TOWARDS THE TUTOR/AID PARADIGM:
DESIGN OF
INTELLIGENT TUTORING SYSTEMS
FOR
OPERATORS OF SUPERVISORY CONTROL SYSTEMS

A THESIS

Presented to

The Academic Faculty

by

Rose Wan-Mui Chu

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Industrial and Systems Engineering

Georgia Institute of Technology

September 1991

TOWARDS THE TUTOR/AID PARADIGM:

DESIGN OF

INTELLIGENT TUTORING SYSTEMS

FOR

OPERATORS OF SUPERVISORY CONTROL SYSTEMS

Approved:

Christine M. Mitchell, Chairperson

T. Govindaraj

Alexander C. Kirlik

James D. Foley

William B. Johnson

Date Approved by Chairperson 9/23/91

DEDICATION

To those I love and cherish the most in life:

... my parents, Ruby Lee and Ching-Lau Chu, for the chance to be the best I can be;

... my brothers, David and William, for their encouragement and support;

... my best friend, Tom Pawlowski, for believing in me and putting up with me;

... my newly found "little" sister, Wanda Billings, for the joy she brings in my darkest moments.

ACKNOWLEDGEMENTS

This research has been supported by NASA Goddard Space Flight Center under contract number NAS5-28575 (Walt Truszkowski, technical monitor) and by NASA Ames Research Center. TAE Plus is a registered trademark of the National Aeronautics and Space Administration.

I would like to express my sincere gratitude to my advisor, Dr. Christine Mitchell, for the opportunity to excel beyond my own expectations, in a research project that has been so challenging and fruitful. My deepest thanks to Patty Jones, the best project partner anyone can ask for, without whom I could not have completed my Ph.D. work. I am especially grateful to Richard Robinson, not only for sharing those late nights with me in correcting computer problems, but also for his friendship and support. To the rest of my committee members, Dr. T. Govindaraj, Dr. Alex Kirlik, Dr. Jim Foley, and Dr. Bill Johnson, I appreciate their interest in my project and their invaluable comments.

I would like to acknowledge the Mission Operations personnel at NASA Goddard Space Flight Center who contributed tremendously to the success of this project. The unconditional support and assistance from the technical, training, and managerial staff made the project all the more enjoyable and meaningful. In particular, a special thanks to Matt Fatig, for being the reliable source of expert knowledge throughout the project, and to Bill Stoffel, for making the evaluation study a reality. I am especially indebted to those who volunteered their time and effort so willingly to participate in the evaluation study. Their enthusiasm definitely made those sleepless nights at NASA bearable, and the experience memorable.

Finally, I wish to express my appreciation for all my colleagues at the Center for Human-Machine Systems Research. Their encouragement and assistance enabled me to survive all these years at Georgia Tech. I will truly treasure their friendship for life. Especially to Kelly Deyoe, for being an inspiration to all of us.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Conventionally, research in computer-based technical training has been approached separately from research in computer-based real-time aiding for complex dynamic systems. This thesis explores the design of an integrated support system that addresses issues in training and aiding concurrently. Specifically, the thesis proposes the tutor/aid paradigm in which an intelligent tutoring system evolves from a tutor to an operator's assistant as the novice operator's expertise increases. The tutor/aid paradigm identifies characteristics of the domain knowledge and student model, and defines a pedagogical structure of an ITS that not only compensates for a novice operator's deficiencies in knowledge and skills, but also prepares the operator to use the tutor as an assistant after training. The pedagogical structure consists of eight lesson types that embody the instructional and knowledge requirements of a novice operator for supervisory control.

This research focuses on the tutoring end of the tutor/aid paradigm by capitalizing on a validated operator's assistant architecture that is based on a prescriptive model of human performance for supervisory control. The goal is to enhance this architecture with intelligent tutoring capabilities. As an initial attempt to explore and illustrate the validity of the tutor/aid paradigm, a proof-of-concept ITS was implemented with the enhanced operator's assistant architecture. GT-VITA (Georgia Tech Visual Inspectable Tutor and Assistant) is a graphical, interactive and real-time ITS developed in the context of the Payload Operations Control Center at NASA Goddard Space Flight Center (GSFC), and evaluated successfully with NASA personnel. Results show that GT-VITA is a flexible and adaptive intelligent tutoring system that successfully trained novice operators to become competent satellite ground controllers. Moreover, reaction from both technical and training personnel at NASA Mission Operations Division was very positive. GT-VITA is being fielded as it is at NASA, and is being integrated with the overall training program for novice satellite ground controllers. Assuming a valid intelligent aiding system, and beyond the success of the intelligent tutoring system reported in this thesis, subsequent research agenda should explore the transition of the tutor to an aid as the next step in validating the tutor/aid paradigm.

VITA

Rose Wan-Mui Chu was born in 1961 in Hong Kong. After graduating from high school in Malaysia, she began college in the University of Montevallo, Alabama, in January, 1980. She eventually received the Bachelor of Industrial Engineering degree from Auburn University, Alabama, in August, 1983. She graduated with highest honor and was also the recipient of the Outstanding Graduate for the School of Engineering. Upon graduation, she began graduate studies in industrial engineering at Auburn University and received the Master of Science degree in March, 1987.

CHAPTER I

INTRODUCTION

With the advancement of computer technology and automation, the trend towards more complex systems has shifted the human operator's responsibilities from low-level system control to high-level supervisory control. Supervisory control involves monitoring and fine-tuning during normal, predominantly automated, system operations, and failure detection, diagnosis and compensation during infrequent but abnormal system events (Sheridan, 1976, Wickens, 1984). The operator must have high-level skills to cope with both normal and abnormal problem solving situations (Rasmussen, 1986). Moreover, abnormal situations in complex systems may involve rare but potentially catastrophic events, so that operator errors may have costly and high risk consequences.

Safety and reliability in complex dynamic systems necessitate the analysis of complexity with reference to the operator's task environment (Rasmussen and Lind, 1981). To reduce task complexity, Leplat (1988) suggests either changing the operator through training or changing the task through redesign. The research project reported in this thesis is concerned with the first option: to prepare a novice operator to cope with task complexity by providing proper training. In fact, Leplat concludes that "the development of a skill may be viewed as the elaboration of a cognitive complexity reducing mechanism" (p.113).

The nature of the supervisory control task and the complexity of the controlled system impose new operator training requirements (Goldstein, 1986). Consequently, traditional training methods such as on-the-job training and classroom instructions are inadequate by themselves. On-the-job training does not accommodate the potentially catastrophic consequences of errors, and it is not conducive to learning some aspects of the control task because of the infrequency of some system events. Although the novice operator may learn about rare events through classroom instruction, such training may not transfer to the actual task

environment since the operator cannot apply the knowledge or practice the skills in realistic (i.e., dynamic) problem solving situations.

Computer technology provides the potential for more effective training tools such as computer-based simulators for complex dynamic systems (Spangenberg, 1976; Goldstein, 1986; Govindaraj, 1987). In general, such simulators are computer-controlled devices designed to train an operator to operate or maintain a system (Kearsley, 1984). There are several reasons for using simulators (Goldstein, 1986). Specifically, a simulator provides a protected environment in which a novice operator can practice supervisory control skills without jeopardizing the performance and safety of the actual system. A simulator can provide a microworld configured repeatedly to reflect scenarios with both normal and abnormal events. A microworld also supports exploratory learning (Papert, 1980). The operator can experiment safely and easily with the simulated system in ways that are neither practical nor safe in the actual system. From a practical standpoint, two factors motivate the use of simulators (and computer-based training systems in general): cost and availability (Johnson, 1988; Kearsley, 1984; Goldstein, 1986). Simulators reduce training costs since they are generally less costly than the actual equipment or systems simulated. In addition, simulators can be used in remote locations where needed and in sufficient numbers to meet high student loads.

Although simulator-based training provides a protected learning environment, the novice operator's learning process is unguided. The simulator does not evaluate the operator's performance or progress. Furthermore, a simulator *per se* does not teach: with only a simulation of the controlled system, it may be difficult for the operator to learn abstract concepts and/or to learn to reason correctly about the system (Woolf, 1986).

ITS and Simulator-Based Training

Recent developments in intelligent tutoring systems (ITS) may offer opportunities to enhance the effectiveness of simulator-based training. While the novice operator is trained within a protected and guided environment, the intelligent tutor can test and evaluate the student's performance and provide context-sensitive help. The intelligent tutor can be *proactive*; it can actively teach difficult and abstract concepts.

The intelligent tutor can be *reactive*; it can guide and assist the student during exploratory learning in a microworld (Wenger, 1987). The intelligent tutor can be *adaptive*; it can tailor simulation scenarios to the student's learning progress. In short, the student receives one-on-one instruction and acquires practical experience in an apprenticeship style learning environment.

ITS may also provide a viable solution to the problem of knowledge attrition in training operators of complex dynamic systems (Bludworth, 1989). Informal on-the-job training and knowledge acquisition by "word of mouth" may mean that some expert knowledge is lost over time as new generations of operators are trained. An ITS enables domain knowledge to be gathered, stabilized and made accessible within a permanent training system for testing and refinement.

Another motivation for the use of ITS in training is practical. Simulator-based training is practiced in relatively few complex domains, mainly in the aviation industry, large power plants and military training (Clymer, 1980). Many simulations, however, are used in large engineering systems for system design and testing. These simulations are often discarded when the system becomes operational. One integrated approach to system design is to allow the re-use of existing simulations for simulator-based training. Specifically, there is an opportunity to incorporate these simulations into an ITS for operator training in large engineering systems.

The Computer-Based Tutor/Assistant Paradigm

This research explores the design of an ITS that evolves from a tutor to an operator's assistant for supervisory control of complex dynamic systems. Not only will the tutor evolve to an assistant during training; ideally, the assistant will eventually function as an operator's aid in the actual task environment.

The computer-based tutor/assistant framework reflects the design philosophy of building a *joint cognitive system* that integrates human and computer intelligence into one system (Woods, 1986a, 1986b). The underlying belief is that, for complex systems, a team integrating both the human and the computer can ultimately achieve the goals of improved system efficiency and safety better than either the human or the computer can achieve alone (Bushman et al., 1989). Based on this belief, the tutor/aid paradigm views

the computer as a cognitive *instrument* as opposed to a cognitive *prosthesis* (Roth et al., 1987) that attempts to improve and amplify human capabilities and not to replace or remedy human deficiencies. In the prosthesis approach, a machine expert is designed to offer advice and problem solutions only. Such a machine expert is inadequate to cope with the unanticipated variability in the problem solving process and may degrade the human's problem solving effectiveness (Roth et al., 1987).

The cognitive instrument approach to the tutor/aid paradigm is problem-driven, i.e., the approach is rooted in the context-specific issues and problems of training technically oriented adults in complex engineering systems (Roth et al., 1987). First, for training in large-scale engineering systems, an ITS that can be used both during and after training is more cost-effective; the initial cost of developing the system can be more easily justified. Second, novice operators may be more motivated to learn and to use the computer training system given that the system which functions as a tutor during training will become an assistant after training. Third, an assistant that functions as a tutor for a novice operator helps to foster operator understanding of the assistant's capabilities and limitations. Many computer-based expert advisors are unsuccessful due to the users' lack of trust in the systems (Woods, 1986a, 1986b). The tutor/aid paradigm proposes a symbiotic relationship between the human operator and the computer system in which the computer system evolves from a tutor to an aid as the operator evolves from a novice to an expert, and thus may be better understood and trusted by the user. In short, the tutor/aid paradigm addresses the issues of computer-based training and real-time computer aiding in one integrated support system for operators of complex domains (Figure 1-1).

The proposed integrated support system, however, is not the sum of a training system and an aiding system. Rather, the transition from training to aiding is imperative to the validity and utility of the tutor/aid paradigm. Thus, the ultimate research agenda is to explore the paradigm from all three avenues: intelligent tutoring systems, real-time intelligent aiding, and the evolution from tutor to aid. Of the three research avenues, only intelligent aiding has been sufficiently explored for complex domains with existing and successful methodologies. The goal of the current research is to focus on the intelligent tutoring aspect, an important piece towards the ultimate understanding of the tutor/aid paradigm.

Given the motivation and goals of the research, Chapter II addresses issues and reviews existing ITSs that are relevant to the technical training of operators in complex domains. Chapter III proposes and discusses in detail the characteristics of an ITS that evolves from a tutor to an operator's aid. Chapter IV further describes the pedagogical design of an ITS -- the heart of the tutor/aid paradigm. Chapter V presents an ITS architecture that enables the proposed tutor/aid characteristics and pedagogical design to be modeled and implemented. This architecture enhances an existing and successful architecture for a computer-based operator's associate that is based on the Operator Function Model, a prescriptive model of human performance in supervisory control (Mitchell, 1987). Thus, the current research capitalizes on a validated operator's associate which represents the final end in the evolution of the tutor to an aid.

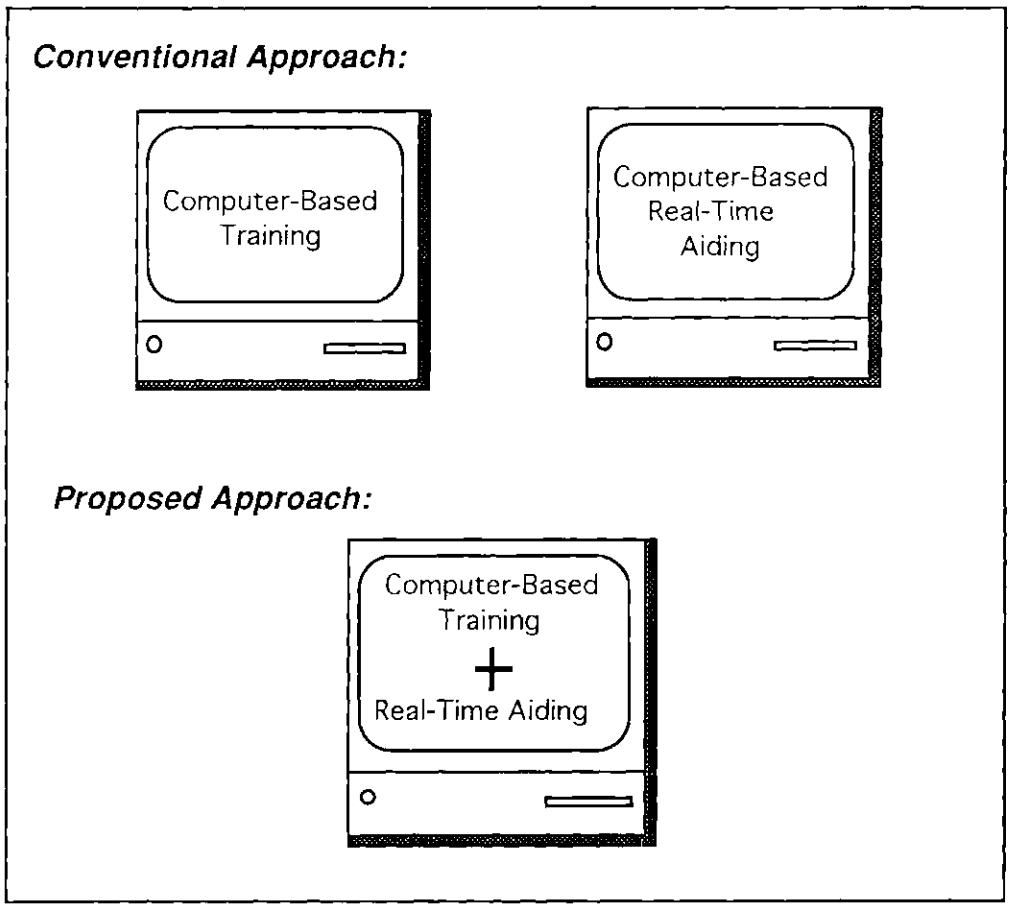


Figure 1-1. The tutor/aid paradigm

As a first step to illustrate the tutor/aid paradigm, a proof-of-concept ITS is implemented and evaluated within a particular supervisory control context. Chapter VI describes the domain of application. The chapter also discusses the process of knowledge engineering and operator modeling that preceded the implementation of the proof-of-concept ITS. Next, Chapters VII, VIII and IX detail the implementation of the ITS for the domain from various perspectives. The ITS interface and its functionalities embody the characteristics and pedagogical design proposed by the tutor/aid paradigm. Chapter X presents the methodology for evaluating the effectiveness of intelligent tutoring aspect of the tutor/aid concept. Problems and issues associated with the methodology are discussed. Chapter XI reports and discusses both qualitative and quantitative results collected from the evaluation process. Finally, Chapter XII ends with some conclusions and proposes areas for future research.

CHAPTER II

TUTORS FOR ADULTS IN COMPLEX DYNAMIC SYSTEMS

This chapter reviews research in intelligent tutoring systems (ITSs) that is relevant to technical training in complex domains. First, the general architecture and dimensions of ITSs are presented. Next, issues that distinguish a *technical training* ITS versus an *educational* ITS are addressed with respect to the target domain and the target student. Finally, design issues for technical training ITSs are further discussed along three dimensions: *communication*, *instruction*, and *knowledge*.

Architecture of Intelligent Tutoring Systems

Over the years, many researchers have surveyed the field of ITS extensively for a wide range of domains, from subtraction skills to medical diagnosis (Sleeman and Brown, 1982; Wenger, 1987; Polson and Richardson, 1988; Mandl and Lesgold, 1987). The general consensus is that an ITS consists of four basic components: domain expertise, student model, pedagogical expertise, and user interface. The domain expertise module represents knowledge to be tutored to the student and functions as a standard for evaluating the student's performance (Wenger, 1987; Fink, 1991). The student model records the student's performance and represents the tutor's hypothesis of the student's state of knowledge. The tutor builds and updates the student model dynamically by diagnosing student actions. The representation and diagnosis of the student model has been termed the *student modeling* problem (VanLehn, 1988). The pedagogical expertise represents "knowledge about how to communicate knowledge" (Wenger, 1987, p. 6). This module is responsible for both curriculum and instruction (Halff, 1988). Finally, the interface serves as the communication channel between the tutor and the student (Burns and Capps, 1988). Recently, the design of

ITS has evolved to include a device or operational simulation of the domain that contributes to the tutor's domain knowledge (Burns and Parlett, 1991). Figure 2-1 shows the general architecture of an ITS.

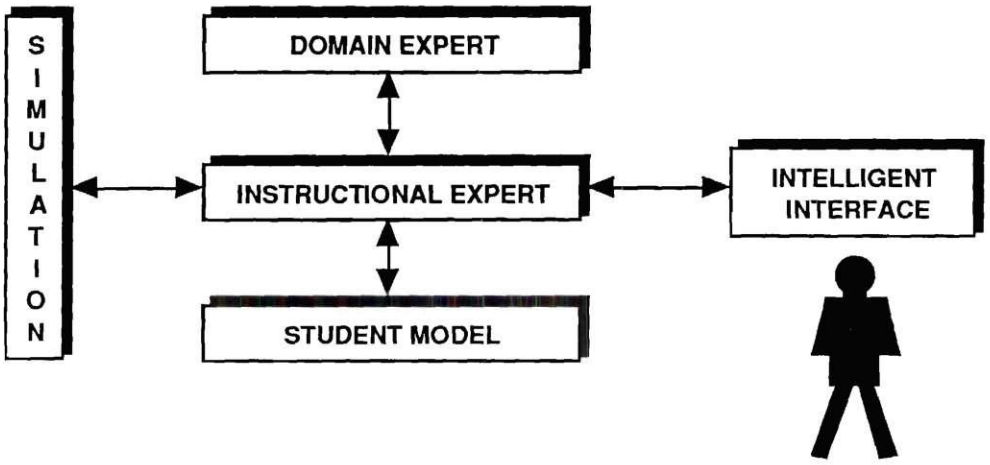


Figure 2-1. General ITS architecture (after Burns and Parlett, 1991)

Perspectives on ITS

Fink (1991) suggests that issues in ITS have been addressed from two major perspectives: psychology and computer science. The psychology perspective stems from research in the fields of psychology, education and human factors that focuses on the instructional issues for the *human learner*. The computer science perspective stems from research in computer science, artificial intelligence and software engineering that focuses on domain issues for the *computer tutor*. For example, Barr and FeigenBaum (1982) review the research as an application and exploration of artificial intelligence techniques in knowledge representation and inference mechanisms. In contrast, Mandl and Lesgold (1988) present recent research in ITS that emphasizes issues in learning and instruction.

From the design and development point of view, Wenger (1987) regards an ITS as a knowledge communications system and emphasizes the interdisciplinary nature of the field. In fact, Burns and Parlett (1991) recognize the interactivity between the various components of an ITS and propose a design perspective that integrates these components within the dimensions of *communication*, *instruction* and *knowledge*. Just as the overall success of an ITS environment depends on both aspects of the human learner

and the computer tutor; the ideal ITS should be designed to capture both the psychology and computer science perspectives (Fink, 1991).

The integrated approach is consistent with the tutor/aid paradigm. As discussed in previous chapter, the tutor/aid paradigm is problem-driven: it is a direct response to the issues of training adults in a technical domain, taking advantage of the available computer technology and tools as a means to capture ideas and concepts about learning and instruction. One of the goals is to effectively train a novice operator to become a competent supervisory controller.

Technical Training ITS versus Educational ITS

A survey of most of the ITS projects shows that the application domains prevalent in the field have been academic subject matters such as geography (e.g., SCHOLAR, Carbonell, 1970), mathematics (e.g., WEST, Burton and Brown, 1982) and computer programming (e.g., PROUST, Johnson and Soloway, 1985). The application of ITS for training in engineering domains such as electronics and power plants has begun only in recent years (e.g., SHERLOCK, Lesgold et al., 1988; AHAB, Fath et al., 1990). Table 2-1 categorizes the various ITSs in terms of their problem domains. In keeping with the goal to design intelligent tutoring systems for supervisory control, it becomes necessary to examine the differences between tutors built for education and tutors built for technical training. These differences can be addressed in terms of the nature of the target domain and the target student.

The Target Domain

Consider the world of algebra and the world of steam propulsion plants. A tutor preparing a student for the former is very different from one for the latter. The differences between the two worlds can be analyzed in terms of their complexity. The notion of complexity here is taken from the vantage point of an individual performing some tasks in a world. Thus, the complexity of the subject matter per se is not undermined. Woods (1988) characterizes a world in four interconnected dimensions: dynamism, the number and extensiveness of interacting parts, uncertainty and risk. A complex dynamic system is a world that

Table 2-1. A summary of major systems in ITS research categorized by domain of application

Domain	System	Reference
Mathematics	EXCHECK BUGGY INTEGRATION Tutor MACSYMA ADVISOR QUADRATIC Tutor WEST WUSOR ALGEBRALAND GEOMETRY Tutor LMS/PIXIE MOTIONS	McDonald, 1981 Burton, 1982 Kimball, 1982 Genesereth, 1982 O' Shea, 1982 Burton & Brown, 1982 Goldstein, 1982 Brown, 1983 Anderson et al., 1985a Sleeman, 1987 Thompson, 1987
Geography	SCHOLAR METEOROLOGY WHY MENO-TUTOR	Carbonell, 1970 Brown et al., 1973 Stevens et al., 1982 Woolf & McDonald, 1984a, 1984b
Programming	BIP FLOW Tutor SPADE MENO-TUTOR LISP Tutor PROUST BRIDGE	Barr et al., 1976 Gentner, 1979 Miller, 1982 Woolf & McDonald, 1984 Reiser et al., 1985 Johnson & Soloway, 1987 Bonar & Cunningham, 1988
Physics	Crane-Boom Tutor	Woolf, 1986
Medical diagnosis	GUIDON	Clancey, 1986, 1987
Pulp and paper process	Recovery Boiler Tutor	Woolf, 1986
Steam propulsion plant	STEAMER	Hollan et al., 1984
Electronics troubleshooting	SOPHIE HAWK MACH-III SHERLOCK	Brown et al., 1982 Massey et al., 1988 Lesgold et al., 1988
Helicopter bladefolding system	IMTS	Towne & Munro, 1988
Satellite ground control	ITSSO	Mitchell & Govindaraj., 1989
Marine power plant	AHAB TURBINIA-VYASA	Fath et al., 1990 Vasandani et al., 1989, Vasandani & Govindaray, 1990; Vasandani, 1991

generally scores high on all dimensions. The complexity of the world directly affects the cognitive demands on the person interacting with the world in a problem solving situation.

Educational tutors for domains such as algebra teaches the student about basic concepts and general procedural skills that are static in nature. That is, elements in the domain are not time-varying. Moreover, educational domains usually involve a finite and deterministic number of elements within a particular subject matter. There is little or no risk involved: student actions or errors do not do jeopardize the world of algebra, for example.

In contrast, industrial domains such as power plants are much more complex and dynamic in nature. The many components in these systems change states over time, react to events (e.g., failures), and affect one another in sometimes unpredictable ways. Operator actions affect system behavior and can result in errors that potentially jeopardize system performance and safety. Thus, an operator must possess very domain-specific knowledge and procedures to supervise and control the system. More importantly, the operator must have adequate cognitive skills (e.g., problem formulation and decision making skills) to cope with the complexity of the system.

The Target Student

Educational tutors differ from training tutors not only in "what to teach", but also in "who" they are teaching. In industry, novice operators who are mature and technically oriented adults, learn and are trained specifically because of their vested interest in job performance, and in most cases, because their job mandates them to do so. Johnson (1988) points out other common characteristics among adult learners. Adult learners want their newly acquired knowledge and/or skills to be immediately applicable to job demands. Adult learners want to be in control over their learning process whenever possible. Adult learners are highly motivated to learn. Adult learners solve problems best by integrating experience and knowledge.

In contrast, whether the students are first-graders or college freshmen, in an academic environment, students often learn for the sake of learning and they usually do not have a job at stake. As a result, issues of motivation and relating knowledge to the "real-world" must be addressed *within* the design of eduacational ITSs. The most recent research in Learning Companion Systems (LCS) for the domain of indefinite

integration (Chan and Baskin, 1990) reflects these design issues. By adding a computer companion to the computer tutor, the idea is to "stimulate student's learning through the process of collaboration and competition" (Frasson and Gauthier, 1990, p. 2).

Whereas for training ITSs, the outcome of learning (and training) is more pertinent. That is, a training ITS must be designed specifically to meet the training needs as perceived by novice operators. The guided and protected environment afforded by an ITS enables the trainee to practice problems and explore the simulated domain system, thus enhancing the transfer of training without jeopardizing the actual system performance. Most training ITSs have all capitalized on the exploratory and practice aspects of ITS. Besides guided exploratory learning, other design considerations that are particularly relevant to technical training of adults include allowance for errors with timely feedbacks, learner-initiated performance feedback and appropriate interface design for adults (Johnson, 1988). These design needs attempt to take into account the adult traits identified earlier. Basically, adult learners are goal-oriented and highly motivated. They seek to be in control and want the chance to exercise their intelligence. Finally, an ITS for adults should also be flexible and adaptable to accommodate individual differences. Individual differences among adult learners reflect their maturity, experience, background and styles of interaction.

As indicated in Table 2-1, there are only a few notable ITSs designed specifically for technical training in complex engineering domains. These systems are reviewed and discussed below with respect to the design dimensions of communication, instruction and knowledge (Burns and Parlett., 1991).

ITS for Complex Domains

It is undeniable that designing and building ITSs is an interdisciplinary endeavor and that an ITS is more than the sum of four or more distinct modules put together (i.e., expert, student, pedagogy, interface and simulation). The design perspective proposed by Burns and Parlett (1991) provides the framework for addressing issues and problems in ITSs as a whole. This perspective places less emphasis on whether each module was implemented in a particular ITS, but focuses on how the system addressed issues in aspects of

communication, instruction and knowledge. Thus, absence of a student model, for instance, does not necessarily devalue the instructional dimension of the overall system.

The following discussion begins with a brief introduction to a few of the existing technical training ITSs. Next, each of the design dimensions is presented and relevant issues reviewed. The methods and approaches employed by some of the training ITSs are also discussed within each dimension. Implications for the tutor/aid paradigm are also addressed.

Review of Existing ITSs

SOPHIE is probably the first intelligent tutoring system designed for troubleshooting electrical circuits (Brown et al., 1982). The SOPHIE project emphasizes the generation of meaningful feedback as the tutor reacts to a student's formulation and hypothesis about troubleshooting via a robust natural language interface. Since SOPHIE, and with the advancement in computer technology, ITSs in complex domains have all involved graphical representations of the task environment.

STEAMER (Hollan et al., 1984) pioneered the use of computer graphics to help students develop accurate conceptual models of a complex system. In STEAMER, the application domain is the operation of steam propulsion plants on large ships. The Recovery Boiler Tutor (Woolf, 1986) utilizes graphical, interactive simulation to provide tools for students to reason about complex processes in pulp and paper mills. One of the most extensive ITS project to date is the IMTS (Intelligent Maintenance Training System, Towne and Munro, 1988) project, which pioneered research in ITS authoring environments. Specifically, IMTS's goal is to provide a generic environment with software tools to create and generate instructional interactions for specific devices in operational maintenance training. IMTS uses a generic diagnostic expert model to support a student's activities in fault diagnosis. The first application domain for IMTS is a helicopter bladefolding system.

Besides IMTS, three other recent projects have addressed various issues in designing ITSs for complex systems. SHERLOCK (Lesgold et al., 1988) is developed to address the problem of training technicians complex electronic troubleshooting skills in the Air Force. SHERLOCK provides a realistic task environment in which trainees practice problems with individualized support and feedback. AHAB

(Fath et al., 1990) teaches students how to troubleshoot failures in a marine power plant. AHAB explicitly represents troubleshooting strategies in a prescriptive model of expert performance. Vyasa (Vasandani, 1991) continues the marine power plant project by defining an elaborate ITS methodology for decomposing and organizing knowledge for effective training of diagnostic problem solving. Vyasa does no impose any troubleshooting strategies; rather, it teaches a student to effectively use various sources of diagnostic knowledge for troubleshooting.

The Human-Machine Communication Dimension

Essentially, the interface of an ITS is the window to the rest of the system. Burns and Parlett (1991) suggests that through this interface, the student can engage in various instructional activities that enhance learning. In other words, these activities provide an environment that facilitate learning (Burton, 1988). Furthermore, the interaction between the student and the tutor on this interface (i.e., "interactivity") is central to a flexible and adaptive tutoring environment (Burns and Parlett, 1991). Consequently, this dimension reflects the interaction between the interface, the pedagogical module and the student (Figure 2-2).

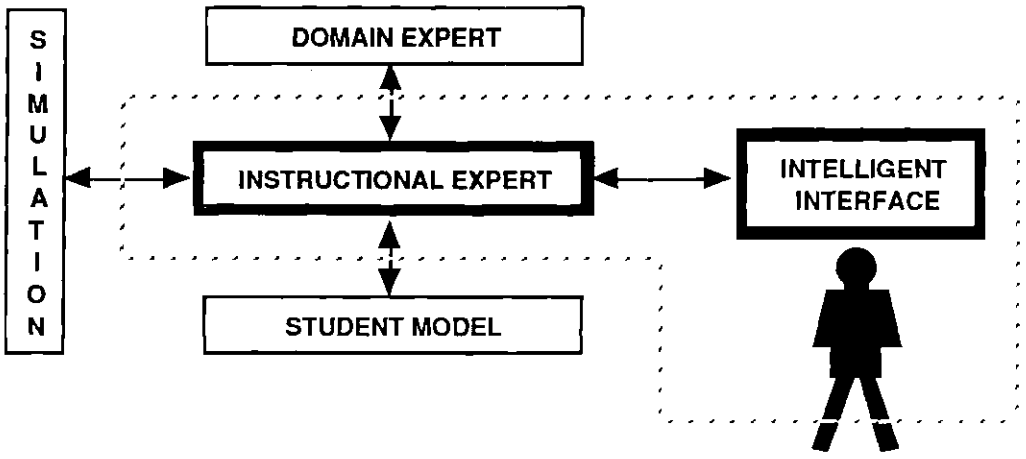


Figure 2-2. The human-machine communication dimension (after Burns and Parlett, 1991)

Swigger (1991) argues that such a "student-centered" tutoring environment requires the computer tutor and the student to communicate as partners in a dynamic interchange of conversational content and context, much like the face-to-face communication between two persons. The conversational content and context are manifested through manipulable objects and operators on the interface that embody the computer's functions and relationships. Swigger suggests that such an interface gives the tutor power and the student the perception of "intelligence".

Other research has also used the face-to-face communication metaphor to study human-computer interactions (Fox, 1988a, 1988b; Suchman, 1987). Fox suggests that tutor-student interactions are a rich diagnostic resource for tutoring and that various conditions of interactions exist for remediation. Suchman suggests that mutual intelligibility must exist for successful communication between humans and machines, just as it is so readily achieved among people. Mutual intelligibility involves not just understanding, but also the ability to repair trouble in understanding when troubles arise. The conversational resources that people depend on and share to achieve mutual intelligibility are consistent with the conversational content and context that exist between two conversational partners, as proposed by Swigger (1991).

With increasing power of computer graphics, interfaces for ITSs are moving away from textual to more graphical mode of display to capture the environment of the domain. Without the advancement in computer technology, the application of ITSs for complex systems could not be realized, as evident by recent ITSs such as STEAMER, IMTS, SHERLOCK, AHAB and VYASA which capitalize on the power of graphics and icons. The interaction with the tutor has also evolved from natural language or command line processing (e.g., SOPHIE) to more direct manipulation interfaces that better embrace the communication partnership. The allowable operations and instructional activities on the interface also reflect the tutoring goals of each system. For example, SOPHIE supports a reactive learning environment where the tutor reacts to the student's hypotheses with feedback and advice. STEAMER encourages the student to explore, inspect and manipulate the graphical simulation objects in developing appropriate mental models of the domain system. In IMTS, SHERLOCK and VYASA, the student learns about fault

diagnosis by querying and manipulating objects on the interface (e.g., gauges) that directly map to objects in the actual task environment.

With respect to the tutor/aid paradigm, the face-to-face communication metaphor is particularly relevant because the relationship between the student and the tutor evolves to that between the operator and the operator's assistant. The tutor/aid proposes a symbiotic relationship between the human operator and the computer support system in complex domains that must be fostered at the communication level.

The Instructional Dimension

Just as in relationships between human tutors and individual students, the goal of an ITS is more than to provide information, but to communicate knowledge, a point maintained by Swigger (1991) and Wenger (1987). Thus, the instructional power of an ITS lies in its abilities to make decisions about what and how much domain knowledge to impart, to hypothesize about the student's progress, and to represent and present the domain knowledge in the most appropriate form (Burns and Parlett, 1991). The instructional power ultimately translates to the type of instructional activities and tools available to the student and the ways the student "sees" and learns about the domain knowledge. That is, the instructional dimension involves the pedagogical module, the student model and the intelligent interface (Figure 2-3).

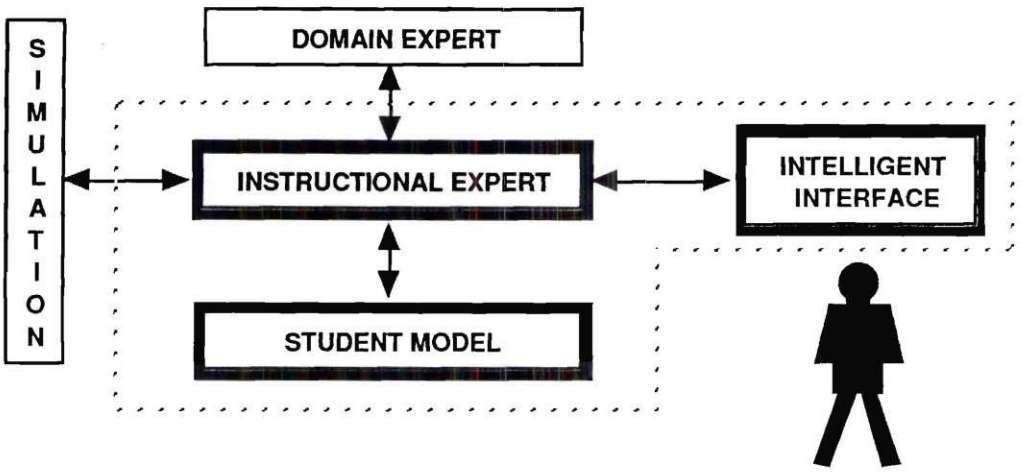


Figure 2-3. The instructional dimension (after Burns and Parlett, 1991)

There is no doubt most researchers will agree that representing teaching knowledge and student modeling are two of the most challenging problems facing the field of ITS. To date, most ITSs do not have a structure that explicitly encodes the pedagogical strategies about curriculum and instruction. Rather, these strategies implicitly guide the tutorial interactions for a specific domain that may or may not generalize to or be reusable for other domains (e.g., WEST). In GUIDON (Clancey, 1987), although teaching knowledge is explicitly represented by tutorial rules, the tutorial module has not been successfully applied even to other medical diagnosis domains (Wenger, 1987). In MENO-TUTOR, tutorial strategies are articulated in a discourse management network that is separate from the domain knowledge (Woolf and McDonald, 1984). However, MENO-TUTOR has only been tested in academic domains of rainfall processes and programming with a very limited natural language interface.

For the class of ITSs of interest here, each capture some underlying pedagogical philosophy, but none has a structure that explicitly encodes the "knowledge to communicate knowledge". For example, IMTS has a set of instructional principles that implicitly drive the tutorial design. SHERLOCK's pedagogical goals are embedded within network nodes that model expert diagnostic activities in a coached practice environment. STEAMER does not instruct; therefore its exploratory mode is evident through the interface alone. The reactive learning environment supported by SOPHIE is captured in the tutorial interactions through the natural language interface. In short, there is a need to identify a pedagogy framework that can capture the teaching knowledge necessary for technical training ITSs. Such a framework forms the heart of the tutor/aid paradigm in characterizing an intelligent tutoring and aiding system.

VanLehn (1988) classifies student models with three dimensions: bandwidth, target knowledge type and student-expert differences. Bandwidth refers to the level of student activities (i.e., mental, intermediate and final) that is available to the tutor for diagnosis. The target knowledge can generally be declarative or procedural. The student-expert differences refer to how the student model represent missing concepts and/or misconceptions. All three dimensions, especially the bandwidth, affect the choice and complexity of the diagnostic strategies used. The difficulty of student modeling problem is further illustrated by Self (1990): the problem encompasses "... representational issues, through plan recognition, mental models, episodic memory to individual differences" (p. 109). However, Self agrees with Sanberg (1987) that a very detailed

student model is not necessary for effective tutoring. From a realistic and practical point of view, Self presents some important suggestions that make the problem of student modeling more manageable:

1. To design student-computer interactions in which the information needed (especially about the student's goals) by the ITS to build a student model are provided naturally by the student while using the ITS, and does not have to be inferred by the ITS from inadequate data.

2. To explicitly link the proposed contents of student models with specific tutorial actions, ideally supported by educational evidence, in order to clarify what is really needed (and not needed) in the student model.

3. To avoid viewing student models solely as devices to support remediation, which is often perceived as implying a behaviorist philosophy of learning and which often cannot be satisfactorily achieved anyway because of various difficulties with the "mal-rule" approach to student modeling.

4. To use student models "constructively" by regarding the contents as representing student beliefs, with no value judgements imposed by the ITS, the ITS's role being to help the student elaborate those beliefs.

5. To make the contents of the student model open to the student, in order to provoke the student to reflect upon its contents and to remove all pretense that the ITS has a perfect understanding of the student (and that ITS designers should build systems which proceed as though they do).

6. To develop ITSs which adopt a more collaborative role, rather than a directive one, for then the style corresponds to a better philosophy of how knowledge is acquired and we do not have to seek such a high degree of fidelity in the student model. (p.121)

Early ITSs for complex domains did not focus on the problem of student modeling and diagnosis. On one hand, the lack of student models could be viewed as a design problem. On the other hand, the interactions afforded by the interfaces in SOPHIE and STEAMER may be sufficient to support the learning environments that the systems intend to provide without the need for student models -- a view consistent with Self's and Sanberg's. For more recent ITSs in the same class, the tutoring goals are more ambitious and thus, student models are needed to diagnose student actions and errors, and to guide subsequent instructional actions (e.g., IMTS, SHERLOCK, AHAB and VYASA).

Some of Self's suggestions for a more manageable student model are particularly relevant with respect to the tutor/aid paradigm. First, good student-computer interactions eliminate "guess-work" by the tutor, thus potentially enhance the perception of the tutor's "intelligence". Second, by making the contents of the student model accessible, mutual intelligibility between the student and the tutor can develop that potentially extends beyond the training phase into the on-line aiding phase. Third, a more collaborative ITS

is consistent with the tutor/aid paradigm to ultimately establish the operator and the computer as a team in ensuring safe and reliable operation of complex dynamic systems.

The Knowledge Dimension

This dimension concerns how "ITS presents the domain to a student" (Burns and Parlett, 1991, p. 8). The student "sees" the domain through the interface which reflects the tutor's expert knowledge base, and the device or operational simulation of the domain (Figure 2-4). Whether device or operational in nature, the simulated environment must exhibit the fidelity of the "real" system at one or more levels: physical (feels the same), display (looks the same), mechanistic (behaves in the same way), conceptual (is thought of as the same) and expert (solves problems the same way as a student) (Burton, 1988).

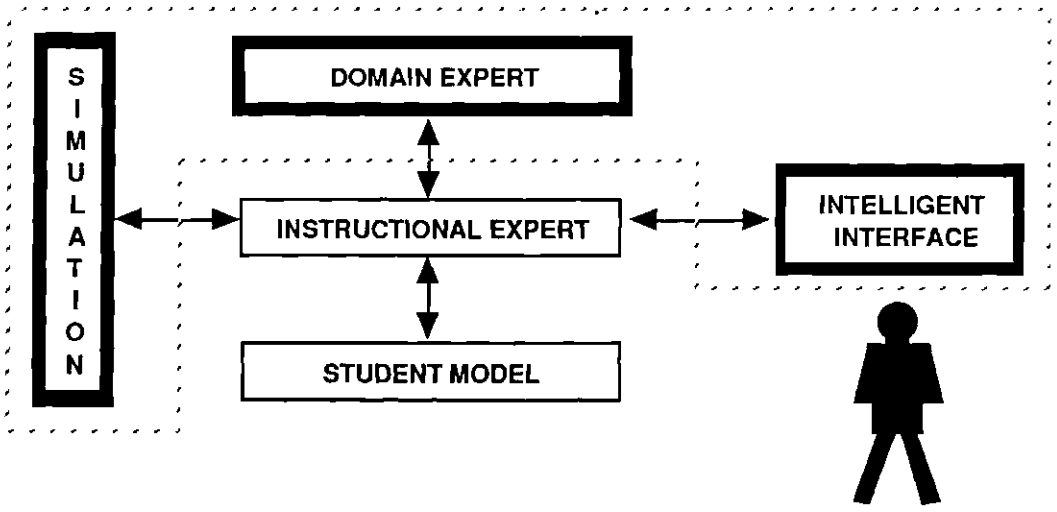


Figure 2-4. The knowledge dimension (after Burns and Parlett, 1991)

The issue of domain knowledge representation is probably the most developed aspect of ITS, benefiting from the research and experiences in artificial intelligence. A tutor's knowledge representation provides a means to impart domain knowledge and a standard for evaluating what a student should know (Fink, 1991). Naturally, the choice of representation depends on the goals of the tutor: what knowledge to teach and what skills to train with respect to the task a student is supposed to learn.

The representation issue is especially critical in *knowledge-rich* domains, where the tutor teaches the conscious and accurate use of knowledge in a high-level problem solving task such as electronic troubleshooting (Fink, 1991). Whereas in *high-performance* domains such as air intercept control, the focus is on speed and automaticity: the tutor trains a student to perform some skills as effortlessly and reliably as possible (Fink, 1991; Regian, 1991). By automatizing some task components, an individual can free up cognitive resources to perform and manage consciously some other functions (Hancock and Pierce, 1984; Schneider, 1985). Moreover, automatic task performance holds up well under stress (Hancock and Pierce, 1984) and is less susceptible to skill degradation (Regian and Schneider, 1986).

Only recently, the distinction between *knowledge-rich* domains and *high performance* domains is introduced to recognize its effects on knowledge representation and pedagogy (Kyllonen and Shute, 1989; Fink, 1991; Regian, 1991). As observed by Fink and Regian, most ITS research to date has emphasized knowledge-rich domains such as physics and medical diagnosis. Even in complex engineering domains such as power plant operations, where both knowledge-rich and high performance task components co-exist (see Woods, 1988), the focus has been the knowledge-rich component, mainly troubleshooting and fault diagnosis (e.g., IMTS, SHERLOCK, AHAB, VYASA). The on-going projects by Fink and Regian represent the first ITS research in high-performance domains. Fink describes a tutor being developed for training mission control console operations at NASA Johnson; Regian describes a prototype tutor for training instrument landing for a simulated F-16 aircraft.

The tutor/aid paradigm is consistent with this recent research: instead of focusing on the knowledge-rich task components, the focus is on designing an ITS to train operations required in supervisory control of complex systems. However, the tutor/aid paradigm goes a step further: the goal is to characterize an ITS that can also evolve to an operator's assistant as the student's expertise increases. Ultimately, issues of tutoring and aiding are integrated in one computer-based intelligent support system.

In the next chapter, the requirements for an intelligent tutoring and aiding system are characterized based on some of the issues relevant to tutors for complex dynamic systems discussed here.

CHAPTER III

THE TUTOR/AID PARADIGM:
CHARACTERISTICS OF A COMPUTER-BASED TUTORING AND AIDING SYSTEM

In order to characterize a computer-based intelligent tutor/aid for a complex dynamic system, it is important to understand the operator's knowledge requirements for effective supervision of the system. Based on the operator's knowledge requirements, the characteristics of the intelligent tutor/aid are discussed in terms of three issues: the representation of domain knowledge, the tutor's pedagogical structure, the student's knowledge representation.

Operator's Knowledge Requirement

The operator of a complex dynamic system needs to have three broad classes of knowledge: declarative knowledge, procedural knowledge (Barr and Feigenbaum, 1981; Anderson, 1988) and operational skill. Declarative knowledge is a general body of facts about the system, its purpose, components, events, and the relations among them. Procedural knowledge emphasizes task procedures to operate the system. Operational skill for supervisory control involves two aspects: cognitive skills and meta-skills. The operator must have the high level cognitive skills to cope with various problem solving situations brought about by the complexity of the system (Woods, 1988). In addition, for effective learning, the novice operator must have meta-skills to monitor and manage his/her own thinking and learning activities (Burton, 1988).

Declarative Knowledge

This class of knowledge encompasses many aspects of a complex system. The operator needs to know *facts* about the system. Also, the operator needs to know the *structure* of the controlled system and its subsystems. The operator also must have *functional* knowledge about what the system does and what his/her responsibilities are. In addition, the operator needs to have *causal* knowledge about reasons for system operations and behavior.

For example, in the domain of NASA satellite ground control, the operator needs to know the various data transmission rates of the spacecraft (facts). The operator needs to know the components that comprise the ground control network and the relationships among the components for different spacecraft events (structure). How and why the different components work, and how they relate to the operator's responsibilities in mission control (functional and causal) are also important.

Procedural Knowledge

This class of knowledge can be viewed as the sequence of instructions that the operator uses to carry out various responsibilities in operating the system. There are procedures specific to a system that an operator must know to monitor normal system operations, to manage failures or abnormalities, and to control certain states of system components. For example, in the NASA satellite ground control system, there are dedicated procedures to prepare for a real-time spacecraft contact and specialized routines to check the spacecraft's health and safety during a contact.

Operational Skill

Operational skill encompasses knowledge at both cognitive and meta levels. At the cognitive level, the operator needs to acquire some high level skills to cope with the demands of a complex task environment in order to properly apply the declarative and procedural knowledge in both normal and abnormal problem solving situations (Woods, 1988). Time management and and cost-benefit analysis are two of the skills necessary for the proper coordination of multiple operator functions in a complex and dynamic environment. Proper coordination is important because system events compete for the operator's

attention. The importance of teaching the coordination (and not just individual execution) of operator tasks is one the many "lessons learned" from the ITSSO (Intelligent Tutoring System for Satellite Ground Operators) project (Mitchell & Govindaraj, 1989). Other crucial skills that result directly from the complex nature of the system include skills in adaptiveness and disturbance management. For example, in the NASA example, although activities for a contact are planned in advanced, the satellite ground controller must know how to coordinate these activities in real time during the contact.

Meta-skills concern knowledge about how to learn effectively (Burton, 1988). The body of *knowledge the novice operator of a complex dynamic system must learn can be overwhelming.* Consequently, at the meta-level, the operator needs to know how to monitor his or her own learning process and manage different activities that aid in learning in order to get the most out of training.

Representation and Communication of Domain Knowledge

The declarative and procedural knowledge together form the domain knowledge that the operator must have. Operational skill can be viewed as the operator's successful acquisition and application of the domain knowledge during training that transfer to the actual task environment. In order to achieve this success, the tutor/aid paradigm proposes characteristics of the domain knowledge for an intelligent tutoring and aiding system that contribute to effective representation and communication. These characteristics are described below.

1. Model of the domain knowledge should be *complete*

In order to communicate the domain knowledge to the novice operator, an intelligent tutor must have a *complete* model of the domain knowledge (Wenger, 1987). Specifically, the tutor must have a model that is at least as rich as the model that the student has at the completion of training. The completeness of domain knowledge, from the student's point of view, provides a context to constrain the otherwise potentially unmanageable amount of knowledge about a complex system that the intelligent tutor needs to consider.

2. Domain knowledge should also be represented in *relevant* and inspectable forms

Domain knowledge that is represented in relevant forms enables the student to inspect and interpret the reasoning steps of the tutor's solution process to a problem (Wenger, 1987). Mathematical equations of system states alone, for example, are irrelevant to an operator in satellite ground control. Particularly for complex domains, it is important that operators can reason about the system in order to cope with the problem of unanticipated variability in system behavior (Roth et al., 1987).

Within the tutor/aid framework, a model of the domain knowledge that is *inspectable* allows the novice operator to begin building a model of the tutor in terms of the tutor's capabilities and limitations, so that after training, the operator can use the tutor (as an aid) effectively. Generally, users are not as willing to use or trust tools that they do not understand. Furthermore, the novice operator may begin developing a model of the domain system that ultimately affects the effectiveness of the operator as a supervisory controller.

3. Domain knowledge should be represented hierarchically

Models of domain knowledge for complex systems must reflect the hierarchic structure of the control and problem solving processes. Operators in complex domains often conceptualize system and operator control tasks hierarchically, and use the hierarchy as a means to organize, reduce and cope with the complexity of the system and the control task (Miller, 1985; Rasmussen, 1986).

The hierarchical representation of knowledge is also important from the perspective of the tutor/aid framework. During training, the operator learns to conceptualize the system in similar ways to those of the tutor; after training, this shared conceptualization of the system can enhance the performance of the human-computer team.

4. The learning environment should support development of high-level skills

Instead of explicitly representing the cognitive skills and meta-skills, the intelligent tutor should provide a learning environment that fosters the development of these skills. Specifically, the tutor should

help the student develop the operational skill to effectively coordinate many concurrent tasks in supervisory control. In practicing the coordination aspect of supervisory control, the student has the opportunity to develop indirectly the cognitive and meta-skills that are necessary for the successful application of domain knowledge.

5. Domain knowledge should be communicated to foster development of useful mental models

The succession of mental models should reflect the student's learning process or evolution of knowledge: incremental learning from simplification to elaboration, from deviation to correction, from abstraction to refinement, and from specialization to generalization (Goldstein, 1980).

6. Domain knowledge should be structured and communicated consistently in an evolution from form/structure to functions and then to operations

The contents of a mental model about a complex system or subsystem consist of both "how-it-works" knowledge and strategic knowledge to use the "how-it-works" knowledge (Kieras, 1988). "How-it-works" knowledge is part of the declarative knowledge that a student should have about the system in terms of the system's structure, form and function. Strategic knowledge consists of procedures and operational skills that a student should know. In a complex domain, the knowledge requirement differs among operators of different system operations. It is very unlikely that any one person in the system organization knows everything about the domain. Thus, the amount and type of "how-it-works" knowledge should be constrained by its suitability in supporting the operator's activities (Kieras, 1988). To foster the development of useful mental models, domain knowledge should be structured consistently in an evolution from form/structure to functions and then to operations. In other words, the student first develops mental models about how the system works followed by mental models about how to operate and control the system.

Table 3-1 shows a summary of the characteristics that pertain to domain knowledge. For the purpose of instruction, an intelligent tutoring system must also have the pedagogical knowledge to communicate

the domain knowledge to novice operators. The tutor/aid proposes a pedagogical structure that captures this knowledge, as discussed below.

Table 3-1. Characteristics of domain knowledge

1. Model of the domain knowledge should be complete
2. Domain knowledge should also be represented in relevant and inspectable forms
3. Domain knowledge should be represented hierarchically
4. The learning environment should support development of high-level skills
5. Domain knowledge should be communicated to foster development of useful mental models
6. Domain knowledge should be structured and communicated consistently in an evolution from form/structure to functions and then to operations

Tutor's Pedagogical Knowledge

The pedagogical structure for the tutor/aid system addresses the issue of "how and when to teach what" to the student operator. The structure varies along two dimensions: *the dimension of knowledge* and *the level of help*. *Declarative, procedural and operational skill* comprise the knowledge dimension. Seven levels of help that an intelligent tutor can provide comprise the help dimension. These levels are: *help, assistance, empowering tools, reactive and exploratory learning, modeling, coaching* and *proactive tutoring*. Burton (1988) proposes this taxonomy to characterize the particular type of help that an intelligent tutoring system offers. Within the tutor/aid paradigm, this classification is extended to identify the levels of help that are appropriate for various facets of tutoring in the dimension of knowledge. The tutor/aid paradigm hypothesizes that the tutor must be capable of providing help on all levels of the spectrum for effective training. Figure 3-1 shows a matrix of pedagogical structure that maps the levels of help to the dimension of knowledge. At the simplest level, the tutor should be able to provide *help* in the form of online system documentation at the student's request throughout the training process. Details of other levels of help for each facet are discussed below.

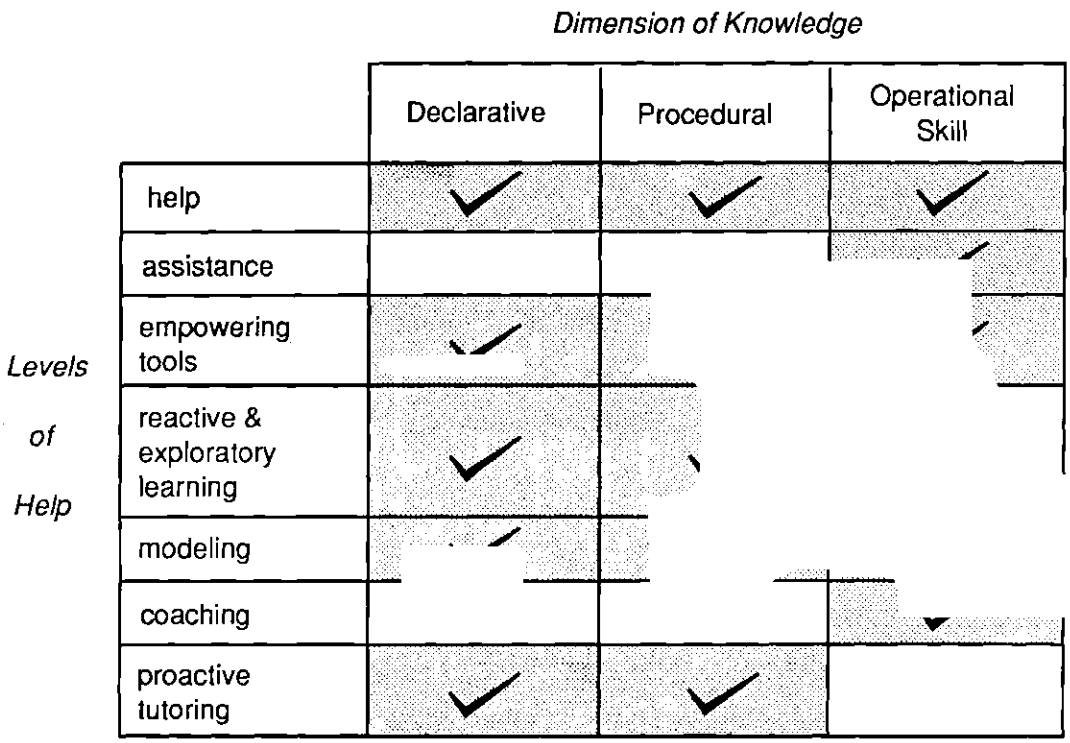


Figure 3-1. Tutor's pedagogical structure

Declarative Tutoring

The philosophy for declarative tutoring is to make a complex domain easier to conceptualize, to make abstract concepts concrete, and to make invisible entities and relations visible. Conceptualization is realized by exploiting graphical capabilities of visualization, animation, and interactiveness. The goal is to help the student develop successively a series of mental models necessary to understand the system and to carry out many aspects of the supervisory control task. The operator must have good mental models (i.e., conceptual representations of the system) to enable reasoning about the system, especially during abnormal or novel situations (Goodstein et al., 1988; Williams et al., 1981).

The tutor's role at this stage is be predominantly *proactive*. The tutor controls the learning process by presenting the domain in visual forms from multiple viewpoints. The student can further learn about the system in terms of the abstract relationships among system components, system behaviors, and system

failures under different circumstances by watching the tutor *model* the effects of different events via animation. In addition, the student can explore and query the system; the tutor at this point is *reactive* to the student's ideas by modeling the effects of these ideas on system behavior and providing meaningful feedback. To facilitate in the student's conceptualization of the domain system, the tutor also provides *empowering tools* that enable the student to explore the tutor's conceptual model of the domain system.

Procedural Tutoring

The procedural tutoring philosophy is to help the student acquire knowledge about *when* to initiate various operational functions, *why* the functions are expected, and *how* to carry them out. Procedural tutoring is implemented in a "learning by doing" environment (Anderson et al., 1985b). The tutor remains *proactive* at this stage, configuring scenarios for the student to learn the operational functions and necessary procedures, and providing guidance to the student when appropriate.

The tutor also has the capability to *model* a control procedure by animating the syntax and effects of each step in the procedure. Thus, before the student actually undertakes a procedure, the student can watch the tutor perform it and see the effects of each activity on system behavior. The tutor provides *empowering tools* such as electronic checklists to track current system states and required functions so that the student can learn to coordinate various functions that demand concurrent attention. Moreover, the tutor supports *exploratory and reactive learning*; the student can experiment with various action sequences and observe their effects on overall system performance.

Operational Skill Tutoring

The philosophy for this facet of tutoring is to promote the development of cognitive skills for effective execution and coordination of multiple concurrent tasks. The concept of "guided discovery learning" (Burton and Brown, 1982) is used to support the philosophy.

In guided discovery learning, the tutor is a *coach*, configuring scenarios for the student to practice his/her knowledge and skills in realistic problem situations, and intervening only when guidance or feedback is appropriate. The student has the option of asking the tutor for *assistance* at any point in a task (i.e., the

tutor can actually assume responsibility for doing parts of the task). This option is important so that the student can concentrate on a crucial but unfamiliar aspect of the problem situation and not be slowed down by routine details (Burton, 1988).

To further enhance learning, the tutor provides *empowering tools* that help the student better conceptualize the system, the problem and potential solution paths. Conceptualization power is necessary to cope with unanticipated variability in complex tasks (Woods, 1987). Presentation aids to reify a system state or a solution process graphically, and on-line comments or critiques on the student's problem solving activities (Burton, 1988; Woods, 1987) are examples of empowering tools that the tutor provides.

Transition from Tutor to Aid

After the training stages, the goal is to encourage the student to make use of the tutor as an aid where appropriate and effective (e.g., to display and track the steps in a procedure), and to highlight those areas in which the student must make his/her own decisions or assessments. The student learns to delegate some operational tasks to the aid within the limits of the aid's capabilities. At this stage, the student is in control of the task scenarios, requesting assistance from the aid only when needed.

In short, the tutor/aid paradigm proposes that the tutor not only compensates for the student's deficiencies in knowledge and skills by providing appropriate levels of help, but also trains the student to use the tutor as an aid.

Tutor's Knowledge of the Student

Effective tutoring relies not only on the tutor's teaching knowledge but also on the tutor's model of the student. Although as argued by Self (1990) and discussed in Chapter II, a detailed student model is *not* necessary to achieve effective tutoring. From a practical viewpoint, especially in light of suggestions proposed by Self (1990), the tutor's knowledge of the student does not have to be explicitly encoded in one module. Rather, the student model can be "distributed" in nature -- the tutor has knowledge of the student with respect to various sources: the student-tutor interactions on the interface, the pedagogical structure and

the domain knowledge structure. The tutor/aid identifies characteristics of the student model that are specifically formulated in the context of complex dynamic systems. These characteristics are described below.

1. The student model should be an overlay, limited-bug model

The student model for the tutor/aid system should be an overlay model (Carr and Goldstein, 1977). An overlay model represents the student's state of knowledge as a subset of the target state of knowledge hypothesized by the tutor. The student model should also include a limited-bug model (Fath et al., 1989) that represents frequently encountered errors or classes of errors in the student's knowledge or operational functions. Thus, student actions can be interpreted as incomplete knowledge via the overlay model, and as incorrect knowledge via the limited-bug model.

2. The diagnostic strategy should be opportunistic

The student model can diagnose opportunistically a student's actions and responses to infer his/her state of knowledge. Specifically, the tutor uses a combination of top-down and bottom-up inference mechanisms. From the top-down perspective, events in a complex dynamic system that unfold in time coupled with specific system requirements and constraints are good predictors of what the student should be doing. From the bottom-up perspective, the student's actions are informative about his or her problem solving behavior in light of the immediate problem solving situations. Moreover, the diagnosis applies to both the student's solution and the solution process.

3. The student model should be inspectable

The tutor allows the student to inspect and query the tutor's current representation of the student's progress. From the perspective of the tutor/aid paradigm, a student model that is inspectable supports the development of mutual intelligibility between the tutor and the student (Suchman, 1987). This joint conceptual model extends after the training process and provides the basis for trust and mutual effectiveness

of the human-computer team as the tutor evolves to an aid. A human-computer team with a shared understanding can potentially result in more system efficiency and safety (Roth et al., 1987).

From the student's standpoint, a student model that is inspectable allows the student to develop a thorough understanding of the tutor's capabilities and limitations in assessing his/her training progress. Moreover, this understanding may be further enhanced when the student is allowed to query and provide feedback about his or her own learning process.

4. The student model should be sensitive to the "situatedness" of a student's action

The student model should be sensitive to the "situatedness" of a student's action in interpreting the action. A student's action is a result of the resources and constraints afforded by the situation's particulars at the moment (Suchman, 1987). Resources are elements in the situation that contribute positively to the execution of an action. Constraints restrict potential actions or compel an action unavoidable at the time. Thus, by considering the constraints and resources that a student has to work with, the tutor can better understand the student's action.

5. The student model should be sensitive to the interactions between the tutor and the student

The student model should be sensitive to the interactions between the tutor and the student in diagnosing the student's progress (Fox, 1988a). The student's response or the lack of it, the manner and timing of the response are all diagnostic information available to the tutor to assess the student's knowledge and to further guide tutorial interactions. Various conditions of interactions are also important to trigger the tutor's remediation of the student's errors (Fox, 1988b). Specifically, when and how an error occurs in the student's problem solving activities has great impact on the student's and the tutor's subsequent responses. For example, when a sequence of student actions is indicative of the student's confusion in arriving at a solution, the tutor initiates the correction and guides the student in remediating the source of the confusion (Fox, 1988b).

6. The student model should account for both the competence and performance of the student

The student model should consist of two sub-models: *competence* model and *performance* model (Lesgold et al., 1988). At the global level, the competence model records the student's progress spanning the entire training period (which consists of several "sessions") in order to identify when the student may have achieved the training goal, i.e., competence as a supervisory controller. At the local level, the tutor models the student performance (i.e., the performance model) from "lesson" to "lesson" within one tutorial session. The performance model guides the tutorial interactions and decisions within sessions.

Table 3-2. Characteristics of student model

1. The student model should be an overlay, limited-bug model
2. The diagnostic strategy should be opportunistic
3. The student model should be inspectable
4. The student model should be sensitive to the "situatedness" of a student's action
5. The student model should be sensitive to the interactions between the tutor and the student
6. The student model should account for both the competence and performance of the student

Table 3-2 lists the characteristics for the student model. In summary, this thesis proposes the tutor/aid paradigm which embodies a set of characteristics of the domain knowledge, the pedagogical knowledge, and knowledge of the student for an intelligent tutoring system that evolves from a tutor to an aid. In particular, the pedagogical matrix provides a framework for identifying the instructional units that the tutor must have to structure the novice operator's training process. This pedagogical framework forms the heart of the tutor/aid paradigm and is described in the next chapter.

CHAPTER IV

THE TUTOR/AID PARADIGM: PEDAGOGICAL DESIGN AND ARCHITECTURE

Chapter III defines the characteristics of an intelligent tutoring and aiding system in terms of the domain knowledge to be communicated, the pedagogical matrix that defines "when and how to teach what", and the tutor's knowledge of the student. Based on the pedagogical matrix, eight concrete *lesson types* are proposed to explicitly capture the knowledge and instructional requirements for training novice operators of complex dynamic systems.

This chapter describes the instructional goal and strategies of each lesson type. Next, an architecture to structure the lesson types is proposed. This architecture provides the framework for representing both *teaching* knowledge and *communication* knowledge, and for overall pedagogical control of tutorial interactions.

Before describing the lesson types, it is helpful to distinguish between system interface and the tutorial interface of an intelligent tutoring system. The system interface represents system components visually and provides the means for the student to carry out operator control activities. The tutorial interface channels communication between the computer tutor and the student within an instructional context.

The pedagogical design proposed here also capitalizes on a real-time simulated task environment over which the tutoring system has control. The tutor should have the ability to specify the creation and timely execution of system events failures (i.e., to schedule events) for effective instruction. The tutor should also have the ability to intervene in the real-time execution of system events. Specifically, the tutor should be able to temporarily halt the simulation clock to capture a particular system state in time, and resume the clock to continue. The functionalities of the event scheduling and timing control in tutoring will become evident in the remainder of this chapter.

The Instructional Units: Lesson Types

This thesis hypothesizes that an intelligent tutoring system must be capable of playing different roles to meet the knowledge and skills demanded on the student, and also to prepare the student to use the tutor as an assistant at the end of training. The mapping between the different levels of help and the dimension of knowledge is illustrated in Figure 4-1. A *lesson type* is the instructional unit that defines the tutoring goals and instructional strategies for one instance of the mapping in the pedagogical matrix. Generally, each lesson type encompasses a particular tutoring role for a particular knowledge requirement. Other levels of help are manifested within each lesson type through instructional activities and tools made available to the student in the learning environment. A total of eight lesson types are identified as shown in Table 4-1.

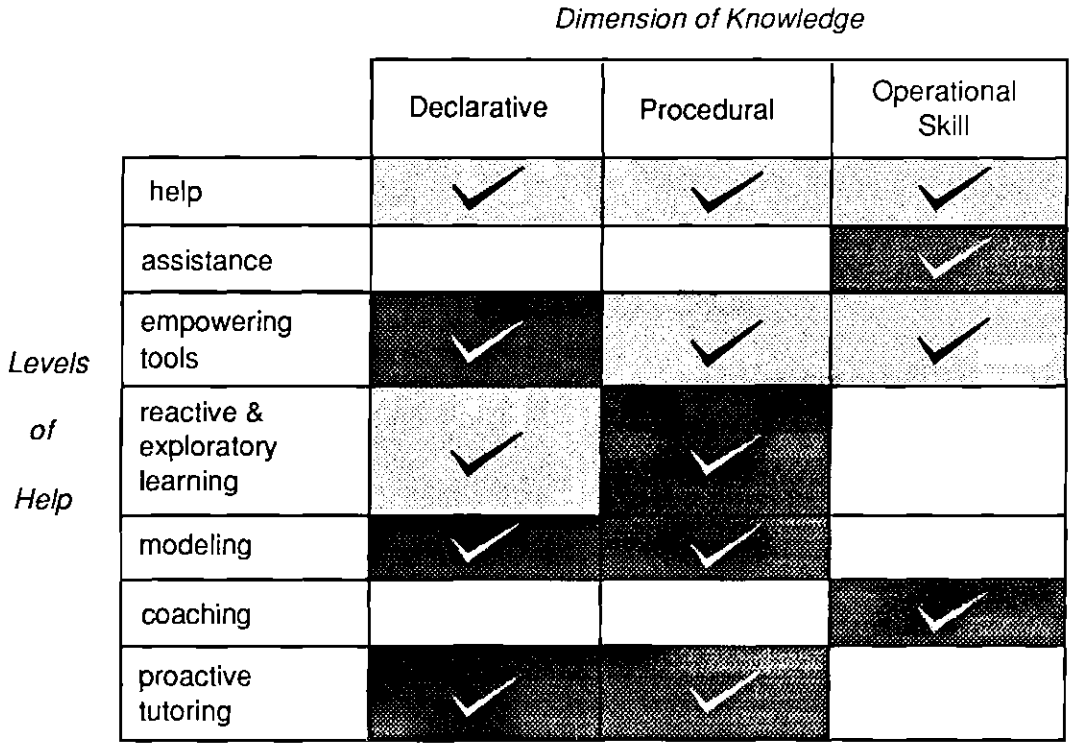


Figure 4-1. Instructional units instanciated for the pedagogical matrix

Table 4-1. The eight pedagogical lesson types

1. Learning System Components
2. Learning System Behavior
3. Exploring Tutor's Knowledge
4. Learning Operations by Example
5. Learning Operations by Doing
6. Practicing Operations with Feedback
7. Practicing Operations with Checkpoints
8. Performing Operations with Tutor as Aid

The eight lesson types represent a reasonable training process that captures the instructional strategies and knowledge requirements to prepare a novice operator to become a competent supervisory controller. Moreover, these lesson types provide a framework for a flexible pedagogical design in a number of ways.

First, the lesson types only specify the *form* of knowledge to be taught and not the *content*; therefore, the instructional content (both knowledge and skills) can be imparted piecewise to the student in multiple lessons of the same lesson type. For intelligent tutors in complex dynamic systems, such flexibility is crucial to help manage the overwhelming amount of information that a student must learn. For instance, lessons for teaching system dynamics can be organized according to categories of system failures.

Second, because the instructional goal is well defined, each lesson type contributes to learning in its own merits. Therefore, the training process does not need to adhere strictly to the order of the eight lessons; rather, the transitions between lesson types can be controlled flexibly by both the tutor and the student based on the student's progress in a lesson and between lessons. From the tutor's perspective, such flexibility accommodates for individual differences in learning. From the student's perspective, such

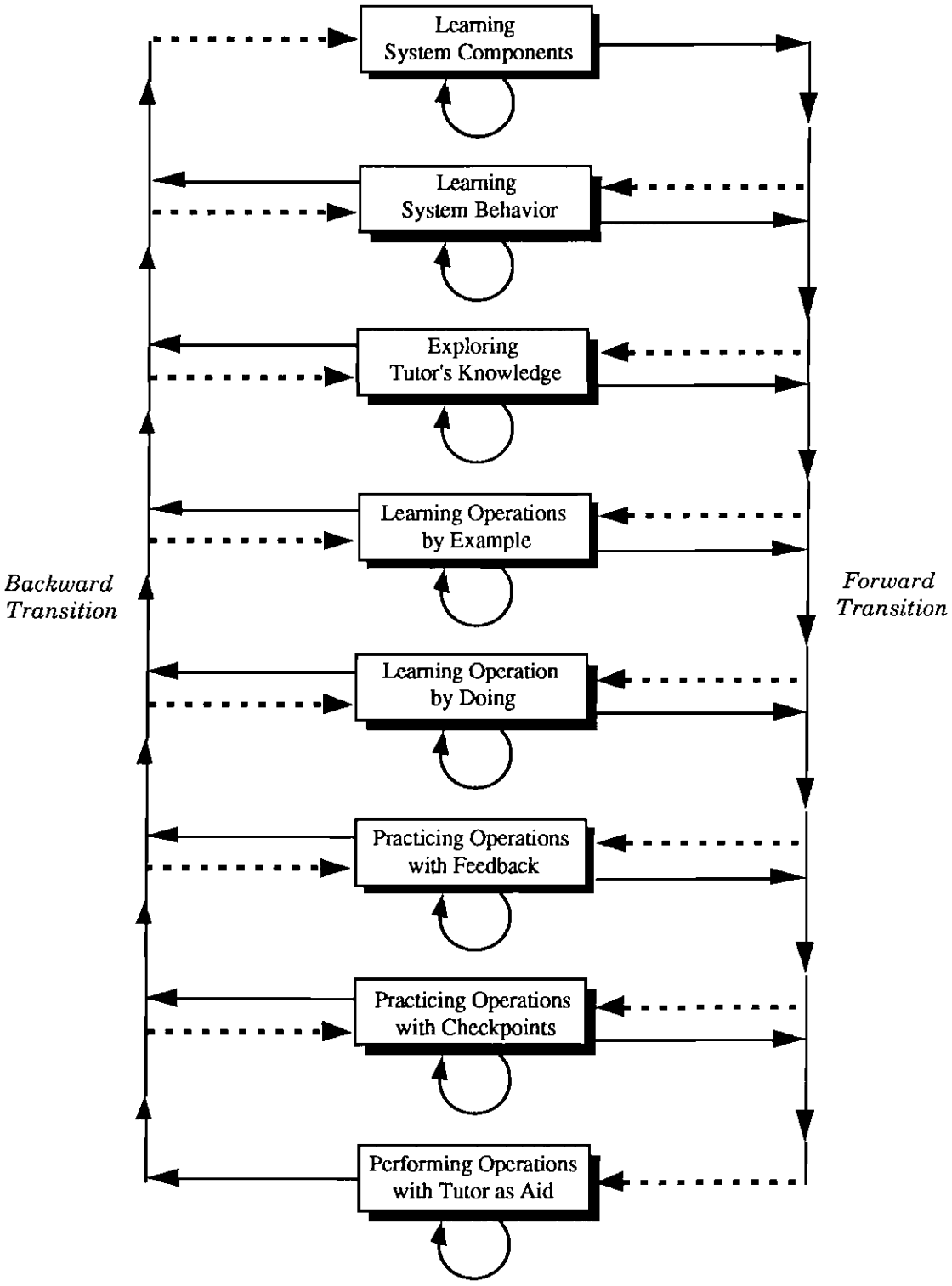


Figure 4-2. The lesson transition network.

flexibility fosters meta-skills to monitor and manage one's own learning process and pace. Figure 4-2 shows the pedagogical network of lesson types. The nodes represent the eight lesson types. The arc between nodes defines the condition for moving from one lesson type to another. A viable future research project is to identify the conditions for transition between lesson types to enhance the design of tutors for complex domains.

Each lesson type is described in detail below in terms of its goals, the instructional strategies to achieve the goals, and an example to illustrate the lesson type. The example is given in the context of a NASA satellite ground control system, the research domain for this thesis. The operator monitors and controls the activities of and communications with an unmanned, near-earth scientific satellite within a NASA network. Details of this application domain will be presented in a later chapter.

Lesson Type 1: Learning System Components (Proactive Declarative)

The goal of this lesson type is to teach the student the body of facts about the domain system. The student acquires the "what" and "why" static knowledge about the domain. In this lesson type, the student becomes familiarized with the system interface that applies both to off-line training and on-line aiding. More importantly, this and subsequent lesson types enable the student to begin developing useful mental models about the domain.

The tutor displays incrementally the visual and graphical representations of various system objects. Each object is annotated with object definition and explanations about its purpose and function. The tutor systematically introduces the system to the student from multiple viewpoints and various levels of abstraction. The student learns about system components and viewpoints that are relevant to the supervisory control of the system.

For example, the tutor displays the overall NASA network and provides an explanation of its functionality. Next, the tutor displays a list of objects representing the components in the NASA network: spacecraft, Tracking and Data Relay Satellite, White Sands ground terminal, and Goddard Space Flight Center.

Lesson Type 2: Learning System Behavior (Modeling Declarative)

The goal of this lesson type is to teach the student about the dynamics of the domain system -- "how" the system behaves. That is, the tutor models system behaviors and relationships among components by animating the effects of various system events and failures in real time. The tutor describes how system behavior and states are represented on the system interface (e.g., color coding schemes). The student also learns about system behavior by exploring various objects and watching the changes in system states. Moreover, by exploring the control activities of various objects, the student begins to develop a sense of where he or she fits in as an operator supervising and controlling the system.

This and subsequent lesson types take advantage of the simulation properties of event scheduling and timing control. First, the tutor shows the failures that are expected for the current scenario configuration, and explains each failure in terms of its symptoms, effects and consequences, corrective operator actions (if any) and its location on the system interface. The student can then learn to locate and recognize the failure when it happens in real time. Second, to further facilitate learning, the tutor encourages the student to pause the scenario which "freezes" a particular system state. The student can observe the system closer and/or uses the chance to "catch up" with system events and tutorial descriptions. The student can restart the scenario at anytime. This timing control empowers the student with the ability to progress through the lesson at his or her own pace, and makes all the knowledge to be acquired during the learning process more manageable.

Although the student is learning about system dynamics, the control activities permissible to the student are also visible on the system interface. In other words, as the student explores various system components, the student also learns about which components are under the operator's commanding power and which are not. However, in order to preserve the lesson's objectives and the scenario configuration, student-initiated control actions are not allowed. Rather, the tutor reacts to the student's control actions with appropriate feedback and explanation of the action without affecting any system states.

From the tutor/aid perspective, the representation of system states and behavior is not only useful during training. The same representation can be helpful in actual task environment for better visualization

and diagnosis of system abnormalities. Ideally, what the student learns in this lesson will carry over after training during on-line operations.

For example, the tutor animates the data flow from the spacecraft to each ground components for a particular scenario configuration. The tutor describes the data flow paths (e.g., color changes) for each NASA subsystem.

Lesson Type 3: Exploring Tutor's Knowledge (Empowering Declarative)

The goal of this lesson type is to allow the student to inspect and explore the tutor's knowledge representations of the domain system. That is, the goal is not to instruct, but to empower the student with the tutor's conceptual model of the domain system in hope that the student can learn to conceptualize the system in ways similar to the tutor. Eventually, this shared conceptualization can potentially enhance the performance of the computer-human team during on-line operations. In this lesson type, the tutor is very unrestrictive, granting the student the freedom to explore the system. The student is encouraged to further study system dynamics and to exercise the option to pause/restart the scenario. In short, learning is reinforced with redundancy and consistency between lessons.

The tutor provides empowering tools for the student to inspect object structures at different levels of abstraction. The tutor's model of each system component's states and behaviors becomes apparent to the student. The student also learns about how the tutor hierarchically organizes these system components. Thus, the tutor communicates the domain knowledge through internal and external representations that map consistently to each other in the graphical or structural modes as selected by the student while the scenario proceeds in real time.

For example, the student can inspect the structure of NASA components such as the spacecraft in various levels of details. The student can also inspect a schematic depicting the NASA object hierarchy to locate the spacecraft's position within the overall hierarchy.

Lesson Type 4: Learning Operations by Example (Modeling Procedural)

The goal of this lesson type is to show the student when to initiate various operational functions, why the functions are expected, and how to carry them out on the controlled system. Specifically, the tutor is playing the role of the operator. The student does not actually undertake the control procedure, but watches the tutor execute the procedure and observes the effects of the activity on system behavior.

The tutor shows the student a set of scheduled commands for the current scenario and explains the intentions for the plan of actions. When it is time for a command to execute (as planned by the tutor), the tutor pauses the scenario and shows the action sequence associated with the command. The tutor also makes available any alternative plans of actions to achieve similar system effects. The student then steps through each action in the action sequence and learns how the tutor carries it out on the system interface. When the student is ready to continue, the student restarts the scenario to watch the consequence of the command just executed. Another way the student monitors the pace of the lesson (and in turns his or her own learning process) is by deciding not to step through the action sequence for a command but to have the tutor execute the command right away. This option is useful for commands that are obvious or repeated; or it may be the case that the student already understands the commands. In between commands, the student is encouraged to explore the system interfaces and study system failures. Again, learning is reinforced.

With respect to the tutor/aid paradigm, this lesson type shows the student the tutor's capabilities and knowledge to undertake operational activities. Long before the tutor evolves to an assistant, the student has the chance to understand the tutor's capabilities and limitations. Ultimately, this understanding translates to the student's trust in the tutor as an assistant which is critical during on-line operations for effective teamwork between the computer and the human. One of the properties of a computer assistant is the ability to carry out a task that has been delegated by the human operator (Rubin et al., 1988).

In the NASA example, periodically, a tape recorder onboard the spacecraft is scheduled to be played back and its data transmitted to earth. The tutor shows the student how to command the spacecraft for a tape recorder playback. The set of commands that the tutor has scheduled to execute is displayed to the student. The student steps through each command as planned by the tutor. The flow of commands from

ground (sent by the tutor) to the spacecraft and the flow of playback data from the spacecraft to the ground are animated.

Lesson Type 5: Learning Operations by Doing (Proactive Procedural)

The goal of this lesson type is to instruct the student when to initiate various operational functions, why the functions are expected, and how to carry them out on the controlled system. The student actually executes the actions for a control procedure as instructed by the tutor. The student then observes the effects of his or her own actions on system behavior.

The strategy for this lesson type is consistent with the previous lesson type. When it is time to complete a command for a particular task, the scenario pauses, and the student requests the action sequence associated with the command. The student then executes each action specified in the sequence. When ready, the student restarts the scenario and observes the consequence of the command just executed. In between procedures, the student is encouraged to explore freely system components and their behavior.

In satellite ground control, before contact (called a "pass") with a spacecraft is acquired, the operator must perform some "pre-pass" activities. When it is time to execute a particular activity, the tutor alerts the student and displays a checklist of commands for it. The tutor marks each command as "done" when the student has correctly executed the steps for the command as suggested.

Lesson Type 6: Practicing Operations with Feedback
(Reactive Procedural to Operational Skill)

The goal of this lesson type is to allow the student to practice previously acquired procedure knowledge in realistic scenario configurations. The student initiates all operator actions while the tutor dynamically monitors these actions (or the lack of them). The tutor does not instruct, but reacts with immediate feedback when the tutor suspects a problem or an error based on the current system states. The tutor's performance assessments of the student are inspectable. Thus, the student knows how to remediate a problem or error and what the tutor's expectations of the student are. Ultimately, the student begins to develop the operational skill to effectively monitor and control the system in a timely manner.

From the student's point of view, the student's goal for lessons of this type is to "compete" with the tutor in the timely execution of various procedures in monitoring the system, and speedy detection and correction of problems or errors. That is, the fewer feedback messages (or "complaints") from the tutor, the better. Thus, this lesson type, when repeated, enables the student to practice and automatize operations within the tutor's expectations.

Because the state of a complex dynamic system varies with time, any assessment on system performance that requires an operator's intervention must be attended to as quickly as possible. Otherwise, the assessment becomes obsolete and/or the magnitude of the problem increases. As a result, even as the student is practicing operations in a realistic context, the tutor pauses the scenario when reacting to a problem (as hypothesized by the tutor). As such, the student is given a chance to understand and correct the problem without any delay. When the tutor's feedback has been attended to, the scenario resumes.

Besides the inspectable student model (in the form of performance assessments), the tutor further supports learning and understanding with these empowering tools: a history of student actions, a history of tutor actions, and a visual, dynamic display of operator activities as hypothesized by the tutor.

From the tutor/aid perspective, an inspectable student model enhances mutual intelligibility between the tutor and the student that in turn can greatly facilitate communication during on-line operations when the tutor becomes the assistant. For example, one minute before a spacecraft contact begins, the tutor assesses the student's performance for proper pre-pass configuration. If an expected action is missing, the tutor alerts the student about the unsatisfactory assessment and pauses the scenario. With the help of the tutor, the student corrects the problem by executing the missing action and the scenario resumes.

Lesson Type 7: Practicing Operations with Checkpoints

(Coach Procedural to Operational Skill)

The goal of this lesson type is the same as the previous lesson type except that the tutor does not react at every error but intervenes if necessary only at cri. cal checkpoints. The idea is to promote further development of cognitive skills for effective coordination and execution of multiple concurrent tasks in a realistic albeit simulated task environment. As a coach, the tutor allows for errors with the belief that the

student will recover from them on his or her own within reasonable time (i.e., before the next checkpoint). In another sense, the coach is accommodating for student's deviations from the tutor's expectations. These deviations may just be a few seconds, for example, or they may reflect individual preferences in ordering some tasks where the order is not significant. To aid in self-recovery, the student is encouraged to inspect the tutor's timely performance assessments of the student's actions. If at the time of the next checkpoint, the suspected problem or error has not been remediated, the coach intervenes.

From the student's point of view, the student's goal here is still to "compete" with the tutor. Specifically, the student attempts to complete a scenario without any interventions from the tutor at the checkpoints. Thus, in this lesson type, the student further practices and automatizes operations, accommodating for error recovery and individual preferences.

Since checkpoints are critical and the domain is characterized by its dynamic nature, the tutor pauses the scenario when intervening to allow the student time to attend to any pending problems or errors that the tutor suggests. When remedial actions have been taken satisfactorily, the scenario resumes.

Just as in the previous lesson, the student has access to a suite of empowering tools that enhance the learning experience: a history of student actions, a history of tutor actions, and a visual, dynamic display of operator activities as hypothesized by the tutor. From the tutor/aid perspective, this lesson type also has an inspectable student model that enhances mutual intelligibility between the tutor and the student.

In a satellite ground control system, the length of a spacecraft contact provides the natural checkpoints for coaching: pre-pass phase, on-pass phase and post-pass phase. The student is responsible for a different set of operator functions at each phase. Unlike the previous example, if the assessment for pre-pass configuration is unsatisfactory, the tutor will not alert the student. Instead, the tutor reassesses the activities for the entire pre-pass phase right before the start of contact. If the student has successfully completed all pre-pass activities, the scenario continues and the student transitions smoothly to the on-pass phase with no interruption. Otherwise, the tutor intervenes and expects the student to attend to all pending actions before the next phase can begin.

Lesson Type 8: Performing Operations with Tutor as an Aid

The goal of this lesson type is to transition the student, now a trained operator, to use the tutor as an aid. The tutor introduces to the student a suite of tools for real-time aiding. The student is encouraged to use these tools in performing various operational activities. Therefore, the success in supervisory control of the domain system depends on the cooperation between the computer assistant and the trained human operator. Research issues for cooperative problem solving are beyond the scope of the tutor/aid paradigm but are addressed by Jones (1991). Jones' work complements the tutor/aid paradigm and focuses on the final end of the evolution from tutor to aid. The transition from the tutor to an aid will be explored in a future research project (Harris, in progress).

General Design Principles

Besides the characteristics and instructional principles discussed so far within the tutor/aid paradigm, there are other general principles that govern the lesson types. These principles are rooted in issues of designing tutors for technically oriented adults. Furthermore, these principles contribute to *meaningful and memorable* interactions between the tutor and the student, and to the apparent *intelligence and personality* of the computer tutor.

1. **The goals and instructions to achieve the goals of a lesson should be available to the student at all times on the tutorial interface.** In this way, the student has a sense of purpose for completing the lesson and that the tutor's expectations are known.
2. **The tutor should encourage reactive exploratory learning where possible.** The tutor's system interface should be designed to naturally support exploratory learning with object-oriented interactive displays. The student should not be afraid of "breaking" the system.
3. **The tutor should encourage self-pacing and self-monitoring of the student's learning process.** In this way, the student is allowed to feel in control. Moreover, by having student initiated events, the tutor can better capture student actions for diagnosis.

4. **All tutorial and interface interactions, and the set of permissible tools between lessons should be consistent and in accordance with lesson goals.** This is to establish the tutor's personality and perceived intelligence so that the student does not encounter "surprises" about how the tutor behaves.
5. **No lesson should end unless the goals have been met.** Regardless of the pace and style of the individual student, the student should always complete a lesson successfully. The sense of accomplishment reinforces and motivates learning.
6. **The system and tutorial interfaces should serve as a continuous external memory to the student.** The interfaces should manifest the knowledge and control operations that the student is learning and practicing through visible objects. In this way, the student does not learn by rote memory alone, but by conceptual understanding.

Pedagogical Architecture

The pedagogical architecture proposed below attempts to encapsulate the characteristics and "behaviors" of the lesson types in a coherent, and flexible structure that is applicable within the domain of complex dynamic systems. This architecture captures teaching knowledge of an intelligent tutoring system as a separate module that interacts extensively with other modules of the system for successful instruction and diagnosis.

Furthermore, the architecture captures the *communication knowledge* necessary in an ITS to manage the interaction between the computer tutor and the student (Swigger, 1991). To establish effective "face-to-face conversation" with the student, Swigger argues that the computer tutor must have (or be) a communication manager to monitor and control the student's learning environment as reflected in the tools and operations available through the interface. The proposed architecture suggests one viable approach to communication management. More importantly, the architecture illustrates the interconnectedness between knowledge to teach and knowledge to communicate. On one hand, what and how a tutor wants to teach

governs the set of operations permissible to the student. On the other hand, the kinds of operations available to the student affect the instructional strategies that the tutor can use. The proposed architecture centers on the idea of a *Lesson Object* which is presented next.

Lesson Objects

The conception of lesson objects for this thesis is largely influenced by Lesgold's (1988) research on curriculum knowledge representation for intelligent tutoring systems. Lesgold suggests that architectures of intelligent instructional systems should be designed in an object-oriented approach which defines".. a set of intelligent fragments of computer program and then orchestrating the interactions among these fragments "(p. 129). Lesgold proposes a lesson object, an intelligent fragment that structures the data and methods to successfully accomplish the instructional goals for which it is responsible.

The current thesis proposes a lesson object structure that is consistent with Lesgold's , but tailored for the lesson types identified within the tutor/aid paradigm. Besides the lesson object, this thesis also proposes a pedagogical object structure that serves as the high level controller for supervising activities of a lesson object.

Lesson Object: Structuring a Lesson Type

The pedagogical architecture proposed here capitalizes on the properties of object-oriented approach to programming. Specifically, the inheritance property enables lesson objects to store common data characteristics and methods through an abstract super class. The abstract class LessonObject represents data such as lesson goals, lesson tutoring mode, lesson knowledge mode, instructions, lesson explanations, and scenario explanations that commonly characterize all lesson types. The abstract class also represents methods to manipulate these data for instruction, such as methods to process a lesson panel and to process various explanation panels.

For each lesson type identified in the pedagogical matrix of the tutor/aid paradigm, a concrete subclass is created. Thus, ProactiveDeclarativeLesson, ModelingDeclarativeLesson, Empowering-DeclarativeLesson, ModelingProceduralLesson, ProactiveProceduralLesson, Reactive-

ProceduralLesson and CoachProceduralLesson are all concrete subclasses of LessonObject that share some common characteristics and methods. Besides inheriting all the data and methods from LessonObject, each subclass contains data and methods dedicated to completing its specific instructional goals with the instructional strategies outlined previously. In general, each subclass has data to keep track of student's performance, status variables to capture the current lesson states and current student states, and concept explanations. Each subclass also contains methods to gather contents of instruction, methods to generate instructional moves, methods to diagnose student actions, and/or methods to transition to next lesson.

One important aspect of the LessonObject structure is the representation and processing of student actions. The basic idea is that as a student interacts with the system and tutorial interfaces, a student action can be categorized as a DIALOG, TARGET, or CONTROL action. A DIALOG action involves a student's interaction with the tutorial interface. A TARGET action involves a student's interaction with the system interface for information request or system visualization. A CONTROL action involves a student's interaction with the system interface for operator control activities. Thus, each lesson subclass has dedicated methods to process the three types of actions according to the corresponding lesson goals. The data and methods for processing the student actions represent the communication knowledge that each lesson object has.

For example, in the NASA example, when a student selects the spacecraft object on the system interface, the outcome of the action is different depending on the current lesson type. For lesson type *Learning System Components*, the associated lesson object processes the student action by displaying declarative explanations about a spacecraft. For lesson type *Learning System Behavior*, the associated lesson object processes the same action by displaying a detailed functional view of the spacecraft subsystems.

While the lesson objects structure both the teaching and communication knowledge, a higher level structure is needed to initiate the appropriate lesson type at the appropriate time, and to integrate the lesson type with the rest of the intelligent tutoring system. Such a structure is described next.

PedagogyModule: Structuring Pedagogical Control

The class PedagogyModule represents the data and methods to oversee the control from lesson to lesson during the training process, and to communicate with other modules of the intelligent tutoring system for domain and diagnostic information. The PedagogyModule object initiates a lesson by creating dynamically an instance of the appropriate LessonObject subclass. This object is also responsible for ending a completed lesson by performing appropriate bookkeeping functions such as updating the lesson history, re-initializing various lesson parameters in preparation of the next lesson. The PedagogyModule contains meta-rules to make decisions about lesson transitions. These meta-rules represent conditions of transition that may be determined by many diagnostic sources about the student's progress and training status. For example, the next lesson type selected may depend on the performance status from a previous lesson and/or on the lesson type specified according to the current lesson goals.

During a lesson, every student action on the system or tutorial interfaces is first processed by the PedagogyModule, and then passed on to the current LessonObject instanciated for further processing. The PedagogyModule also coordinates all messages from other modules of the intelligent tutoring system and filters these messages to the appropriate lesson object dynamically.

In summary, the lesson object provides the structure for representing teaching knowledge of an intelligent tutoring system as defined by lesson types proposed in this chapter. The lesson object also represents communication knowledge which defines the constraints and resources available to the student in interacting with the system and the tutor during a lesson. For effective control of lesson objects and overall integration of the intelligent tutoring system, a pedagogy module object acts as the supervisor of the entire training process. Figure 4-3 depicts the relationship between the pedagogy module and lesson objects.

In the next chapter, an architecture for coordinating the pedagogy module and the lesson objects with the other aspects of an intelligent tutoring and aiding system is presented. This proposed ITS architecture enables the tenets of the tutor/aid paradigm to be implemented and evaluated.

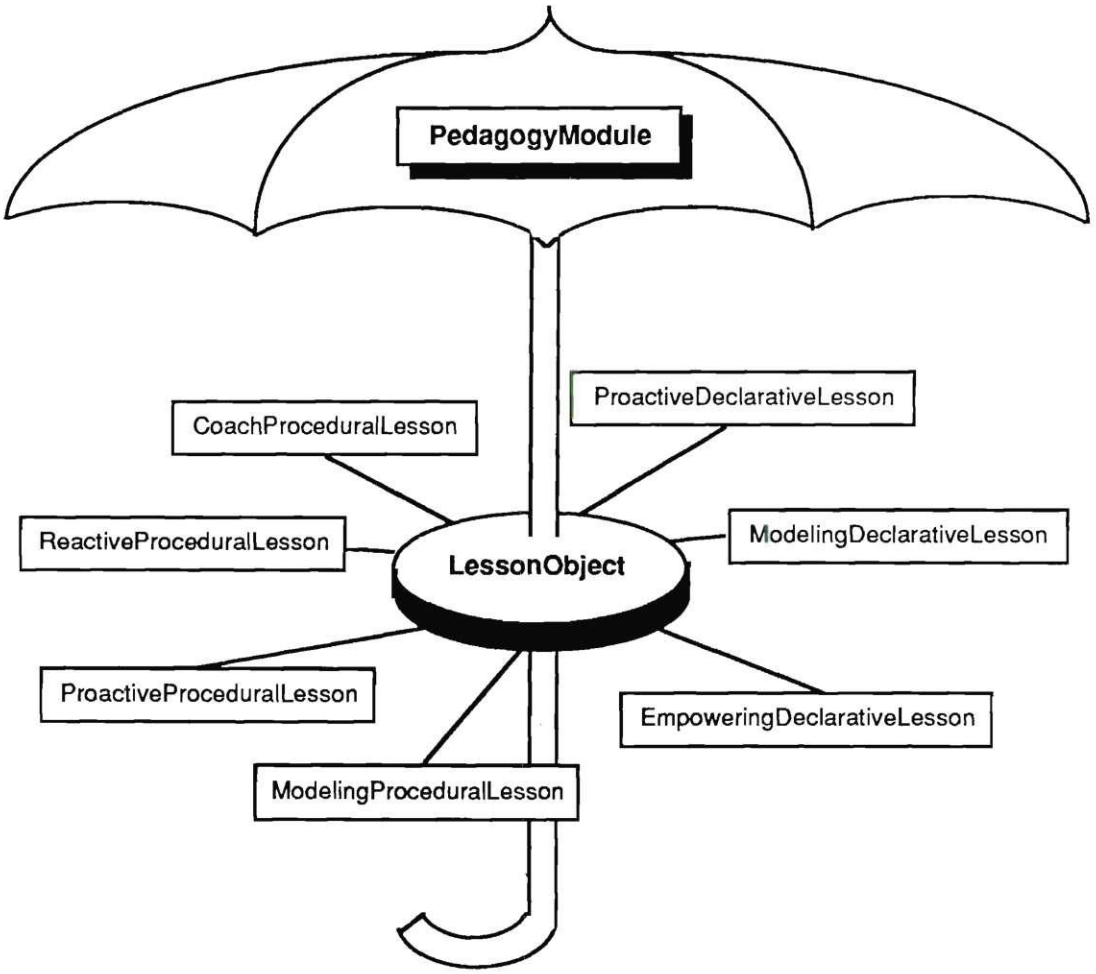


Figure 4-3. The pedagogy module and lesson objects

CHAPTER V

THE TUTOR/AID PARADIGM: IMPLEMENTATION ARCHITECTURE

The proposed characteristics of an intelligent tutoring and aiding system, and the pedagogical design of such a system form the *theory* of the tutor/aid paradigm. In order to test the validity and utility of the theory, the necessary *architecture* to model the theory is addressed in this chapter. First, a general system architecture for the tutor/aid paradigm is discussed. Second, a candidate implementation architecture is presented. Third, the enhanced architecture for the tutor/aid paradigm is proposed.

ITS Architecture

Figure 5-1 shows an ITS architecture tailored to simulation-based training in complex dynamic systems. A dynamic simulator is responsible for creating the simulated task environment. The ITS is "added on" to the simulated environment. The student interacts with both the controlled system and the tutoring system via two separate interfaces. The expert module represents the knowledge to be tutored to the student and functions as a standard for evaluating the student's performance. The student model records the student's performance and represents the tutor's hypothesis of the student's state of knowledge. The pedagogy module represents teaching knowledge.

The tutor/aid paradigm proposes that an ITS for a complex dynamic system should be designed within a distributed environment. A distributed system (versus one in which the tutoring component is embedded within the controlled system) is important for several reasons. First, as the tutor evolves into an assistant, the simulated task environment can be replaced by the actual system. As a result, the tutor/aid system should be a stand-alone system that can be used either with the dynamic simulator during training or

with the actual system after training. Second, from a practical standpoint, a distributed architecture does not hinder the initial design and engineering of the controlled system: decisions about the task environment are made separately from decisions about the training program. Third, a distributed environment preserves the fidelity of the operator's task environment. The operator is in control; the tutor/aid is utilized at the operator's discretion.

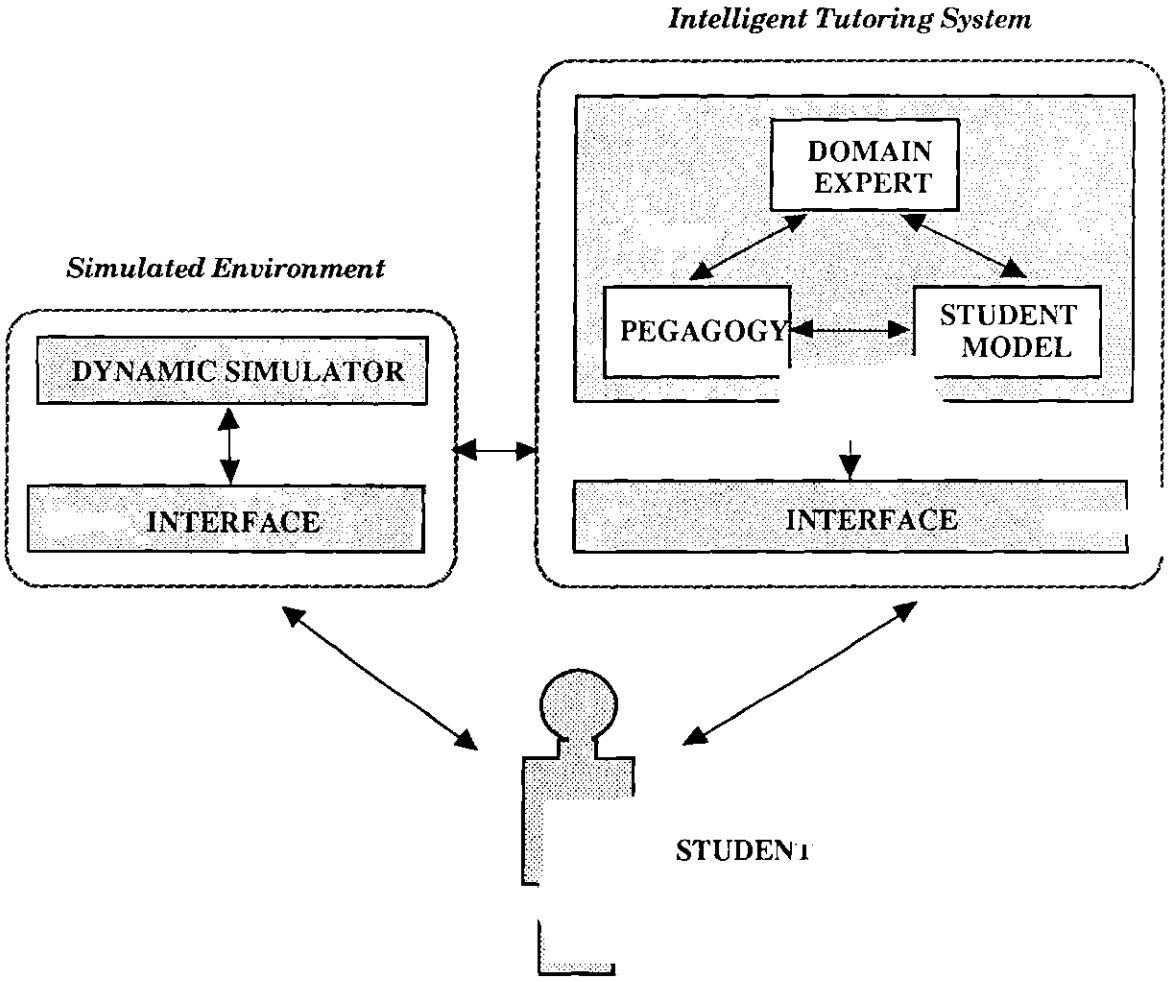


Figure 5-1. ITS architecture for complex dynamic systems (after Burns and Capps, 1988)

Tutor/Aid Architecture

The implementation of the tutor/aid concept capitalizes on the emerging research on operator associates. Specifically, the goal is to extend OFMspert (Operator Function Model Expert System), a generic architecture for a computer-based associate (Rubin et al, 1988), to include an intelligent tutor that evolves into an operator's assistant. The design of OFMspert is consistent with that of the tutor/aid paradigm. OFMspert is an architecture for a cognitive instrument that enhances the effectiveness of the human supervisory controller in the decision making process and represents the final end in the evolution of the tutor to an aid.

OFMspert provides a good beginning for the ITS design because the architecture has structures to represent operator's intentions and system states. These structures are important in order to provide context-sensitive and timely assistance (such as advice and reminders) and to allow the operator to delegate portions of the supervisory control task when needed (Rubin et al., 1988). OFMspert has been tailored to GT-MSOCC (Georgia Tech Multi-Satellite Operations Control Center), a real-time, interactive simulation of MSOCC, NASA's satellite ground control station. The understanding property in GT-MSOCC OFMspert has been rigorously validated by Jones et al. (1990). ALLY, a computer-based associate based on the GT-MSOCC OFMspert architecture augmented with control capabilities, has been designed and tested recently by Bushman (1989). Experimental results showed that a human-ALLY team performed as well as, and in some cases better than, a human-human team in controlling GT-MSOCC.

Consequently, given a valid and proven architecture of a computer-based associate for supervisory control systems, the goal of this research is to design an intelligent tutoring system that capitalizes on the properties and capabilities of OFMspert. Philosophically, when the integrated, intelligent support system plays the role of the tutor, it not only compensates for the operator's deficiencies in knowledge and skills, but it also prepares the operator to eventually use OFMspert as an associate.

The success of OFMspert relies on a sound model of the human operator. This model is a prerequisite for OFMspert's ability to infer operator intentions. The operator model underlying OFMspert

and the OFMspert architecture are summarized below. In the next section, an enhanced OFMspert for the tutor/aid paradigm is presented.

Operator Function Model

The operator function model (OFM) (Mitchell, 1987), a prescriptive model of human performance in supervisory control, defines the knowledge OFMspert needs to perform intent inferencing. An OFM represents the decomposition and coordination of operator activities in the control of a complex dynamic system. The model represents the interrelations between dynamic system states and operator activities in a heterarchical and hierarchical network of finite-state automata. Figure 5-2 shows a generic operator function model.

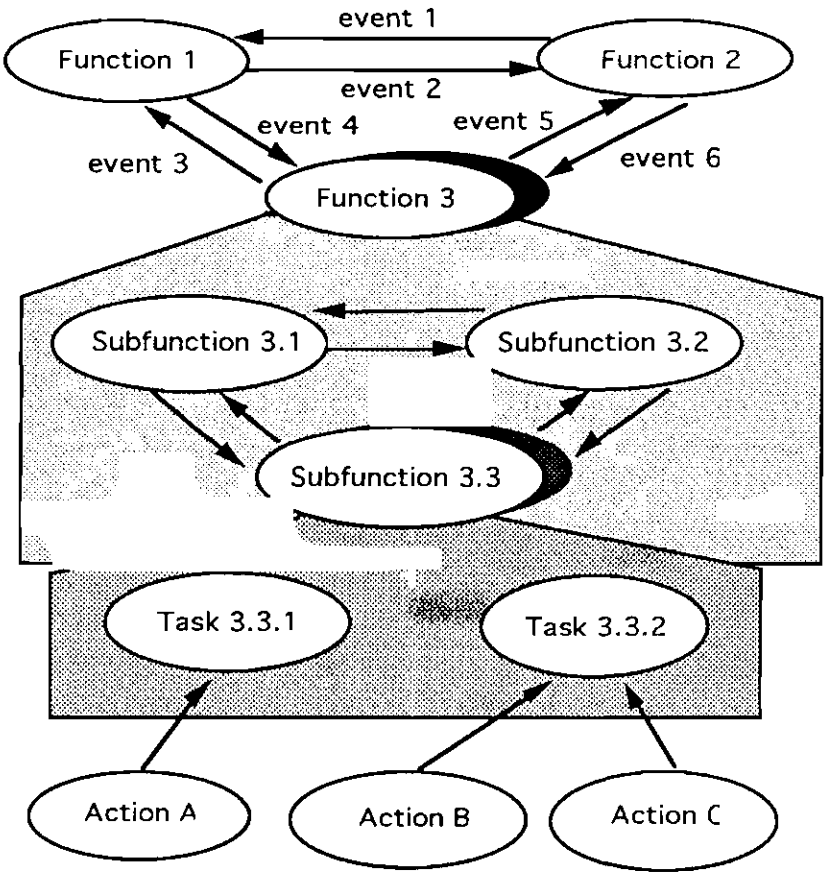


Figure 5-2. The Operator Function Model

Major operator functions are captured at the heterarchic level as network nodes. The heterarchy accounts for the coordination and concurrency of multiple operator activities. Each major operator function is further organized as a hierarchy of network nodes, with typical decomposition of function to component subfunctions, subfunction to component tasks, and task to component actions (both cognitive and manual). The dynamic interaction between the operator and the controlled system is captured in the network arcs between nodes. These arcs represent system triggering events or the results of operator actions that dynamically shifts the operator's focus of attention to different activity node and/or level in the OFM network.

The OFMspert Architecture

OFMspert is a software architecture for implementing an operator's associate in supervisory control systems. Basically, the architecture uses the operator function model to represent knowledge about the controlled system and plausible operator activities, and the blackboard model of problem solving (Nii, 1986) to dynamically hypothesize about operation intentions. Figure 5-3 shows a generic OFMspert architecture with its major components.

OFMspert is designed to be a distributed operator's associate that communicates with the controlled system and the human operator through the *OFMspert Interface*. Thus, the human operator interacts with the controlled system through a task interface, and with OFMspert through the OFMspert Interface.

OFMspert's knowledge of the controlled system and its associated operator activities is encapsulated in three components: *Enhanced Normative Model, State Space* and *Control Environment*. The Enhanced Normative Model represents the prescribed operator function model of functions, subfunctions and tasks as activity trees. This component also contains procedural knowledge that defines OFMspert's control property. The State Space contains OFMspert's knowledge of the controlled system in term of characteristics and current status of its components. The Control Environment reflects the current state of the interface between the operator and the controlled system. With this component, OFMspert can infer operator actions by the type of information that is currently displayed to the operator even when explicit information requests, for example, are not detected.

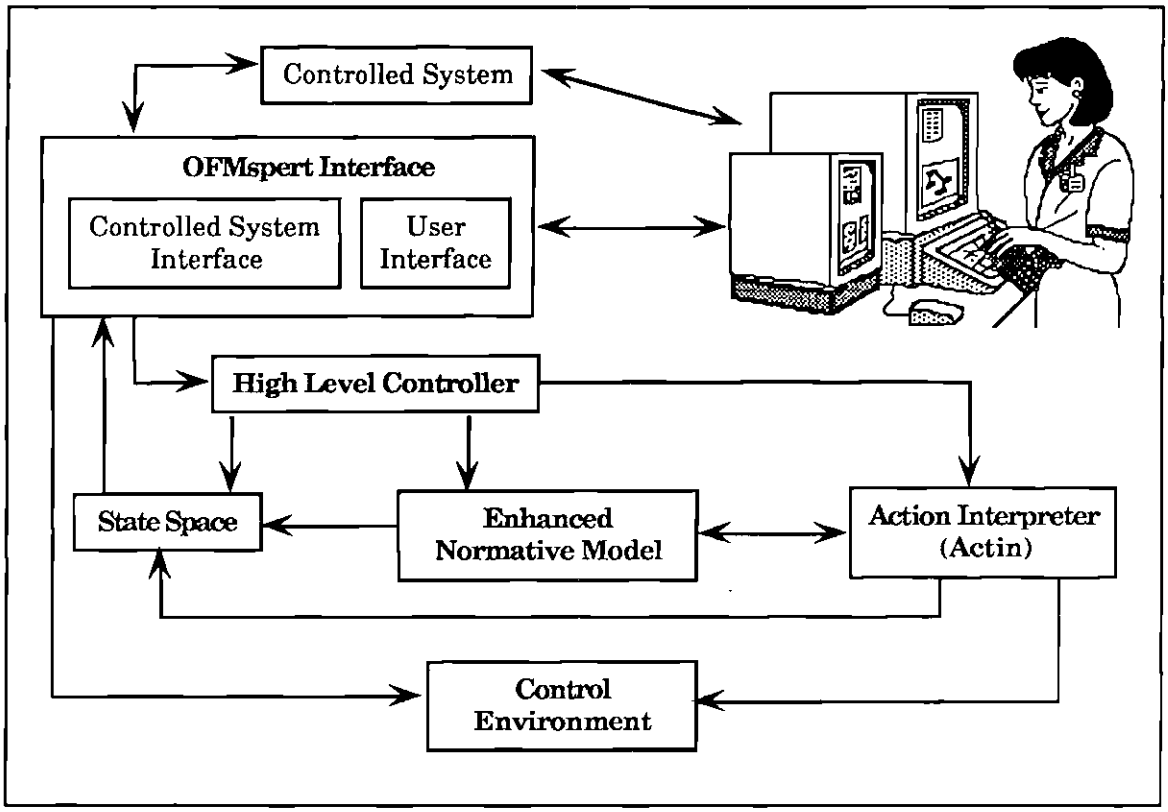


Figure 5-3. The generic OFMspert architecture

The core of OFMspert resides in *ACTIN*, the action interpreter that is responsible for the dynamic intent inferencing capability of OFMspert. ACTIN consists of a blackboard data structure that represents an evolving model of operator intentions. The model of intentions captures OFMspert's current best hypothesis of the functions, subfunctions and tasks (i.e., activity trees) that the operator is attending to, and how well operator actions support the hypothesis (Figure 5-4). Specifically, based on system triggering events, ACTIN posts new activity trees on the blackboard. As operator actions occur, ACTIN posts them on the blackboard and attempts to 'connect' them to any current tasks which the actions support. This process of connection is intent inferencing. The construction, maintenance and assessment of the blackboard data structure are carried out by a collection of knowledge sources within ACTIN.

Finally, the *High Level Controller* acts as OFMspert's events manager and scheduler. This component coordinates timely updates to the State Space, the Control Environment and ACTIN in accordance to triggering system events and operator actions.

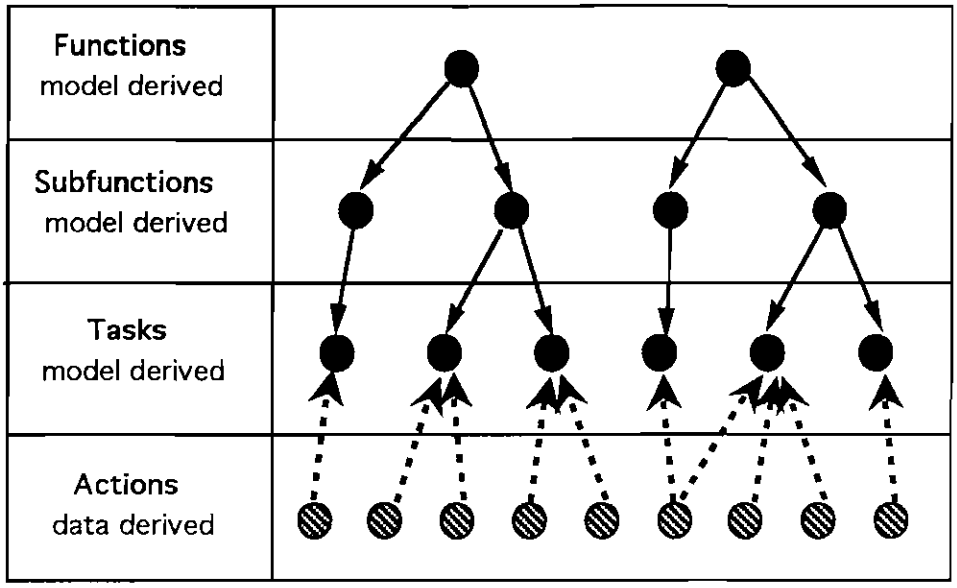


Figure 5-4. ACTIN's intent inferencing structure

Given the basic understanding of OFM and OFMspert, the next step is to enhance the OFMspert's architecture for the tutor/aid paradigm. The following section discusses this enhanced architecture.

OFMspert for Intelligent Tutoring

With respect to the tutor/aid approach, OFMspert represents the final end in the evolution of the tutor to an aid. From a design perspective, the objective becomes one of extending the capabilities of the operator's associate to include intelligent tutoring. In particular, the OFMspert architecture must be analyzed in terms of the characteristics proposed by the tutor/aid paradigm to determine the extent to which OFMspert displays these characteristics (Refer to Table 3-1, Figure 3-1 and Table 3-2 respectively). Consequently, the role of OFMspert within the tutor/aid approach is discussed below with respect to the representation of domain knowledge, the tutor's pedagogical structure, and the tutor's knowledge of the student. Next, based on this analysis, an enhanced OFMspert architecture is formulated.

Representation of Domain Knowledge

OFMspert's knowledge of the domain is distributed in the State Space, Enhanced Normative Model and the Control Environment. The State Space represents states and behaviors of system components. The Enhanced Normative Model consists of the operation function model of operator activities based on system states. The Control Environment represents the interface configuration between the human operator and the controlled system. With respect to the tutor/aid paradigm, OFMspert does have a complete model of domain knowledge that captures the performance of the human operator and the task environment. Also, the hierarchical nature of operator activities is reflected in the operator function model. The State Space can represent the hierarchical organization of system components and their relations.

The complete and hierarchical domain knowledge of OFMspert is sufficient for intelligent aiding of trained human operators. However, to support intelligent tutoring, the knowledge in the State Space and the Enhanced Normative Model are not encoded in *relevant* form. Specifically, OFMspert must be augmented with explicit encoding of declarative knowledge about system components and their characteristics, and of procedural knowledge about control activities in a form that facilitate learning. Right now, OFMspert uses the interface configuration in the Control Environment for better intent inferencing of operator actions. However, for the purpose of instruction, the Control Environment should also capture the characteristics of and relations between interface elements (e.g., element type and parent window).

Tutor's Pedagogical Structure

Although OFMspert is not designed for intelligent tutoring, some of its features are directly applicable to and consistent with the pedagogical structure within the tutor/aid paradigm. The following discussion analyzes OFMspert in terms of the seven *levels of help* suggested by the pedagogical matrix of Figure 3-1.

As a computer-based associate, OFMspert can offer advice, suggestions and/or reminders. The operator can also delegate all or part of the control function to the associate. In other words, OFMspert already has the capabilities to provide *help* and *assistance*. With the OFM, OFMspert has a normative model of a trained human operator. In addition, OFMspert has knowledge about current system states, and

the operator's interaction with the system's controls and displays. That is, OFMspert has the knowledge necessary for *modeling* (or demonstrating the effects of) system events and operator activities on system behavior. As a computer associate, OFMspert is capable of displaying the blackboard data structure graphically which serves as an *empowering tool* for the operator to reason about OFMspert's expectations and hypothesis of system states and operator activities. Other empowering tools such as checklists can be obtained from other knowledge-rich structures within OFMspert. In short, OFMspert, when used during training, has the knowledge to provide *help, assistance, modeling* and *empowering tools* to the novice operator. However, OFMspert does need to be augmented with teaching knowledge to appropriately construct and present the various levels of help in a timely manner within an instructional environment.

OFMspert also lacks the teaching knowledge to be *proactive, reactive* and to be a *coach*. To be *proactive* and *reactive*, the tutor must know when and how to configure and modify system views and/or problem scenarios in response to the student's actions during training. For example, a student must show competence in knowledge about the system elements in terms of form, structure and function before problems that teach operations are presented. To be a *coach*, the tutor must have knowledge about various conditions for interventions: when to interrupt to provide guidance or make suggestions. For example, when a long inactive period after the student executes a procedure is detected, the tutor may intervene and suggest other activities to carry out.

Tutor's Knowledge of the Student

Within the tutor/aid paradigm, the focus is on a distributed student model that captures both the local performance and global competence of the student. The various knowledge-rich structures of OFMspert (i.e., the State Space, the Enhanced Normative Model, the Control Environment, and ACTIN) are important diagnostic resources for enhancing the intelligent tutor's knowledge of the student's progress. In fact, when the student is practicing control activities, the intelligent tutor can take full advantage of OFMspert's dynamic intent inferencing capability for instruction. In particular, OFMspert needs to be supplemented with teaching knowledge to process different diagnostic information (e.g., blackboard assessments) and

record student actions. Ideally, OFMspert should be augmented with a limited-bug library of common errors/problems to increase the tutor's diagnostic power.

The graphical display of ACTIN in OFMspert can be made interactive to serve as a partial inspectable student model. As a tutor, OFMspert's evaluation methods and assessment results should be transparent to the student during the learning process.

Enhanced OFMspert Architecture

Figure 5-5 shows the enhanced OFMspert architecture for intelligent tutoring. There are four areas in which OFMspert is enhanced: the OFMspert interface, domain knowledge, teaching knowledge, and knowledge of the student.

The OFMspert Interface. For intelligent tutoring, OFMspert interfaces not with the actual controlled system, but with a simulated version. To facilitate learning, OFMspert is enhanced to provide an interactive graphical representation of the domain system. Thus, a student learns to interact with the controlled system indirectly through the graphical interface before learning to do so through the actual (albeit simulated) task interface. In other words, OFMspert's interface with the student consists of two components: the system component and the tutorial dialog component. The system component enables the student to interact with the simulated controlled system. The tutorial component channels communication between OFMspert (i.e., the tutor) and the student. Ultimately, the graphical support provided during training is also available to the operator during online operations with the actual system.

OFMspert's Domain Knowledge. The State Space and Enhanced Normative Model are supplemented with explicit representation of declarative and procedural knowledge. Also, the Control Environment is enhanced with characteristics of interface objects and windows to support instruction.

OFMspert's Teaching Knowledge. One major component that has been added to the OFMspert architecture is the Pedagogy Module. The Pedagogy Module captures explicitly the teaching capabilities that OFMspert lacks in one coherent object. The Pedagogy Module preserves the design architecture proposed in Chapter IV that provides the control and processing of instructional units structured as lesson objects. Thus, the various levels of help and instructional strategies are manifested within the

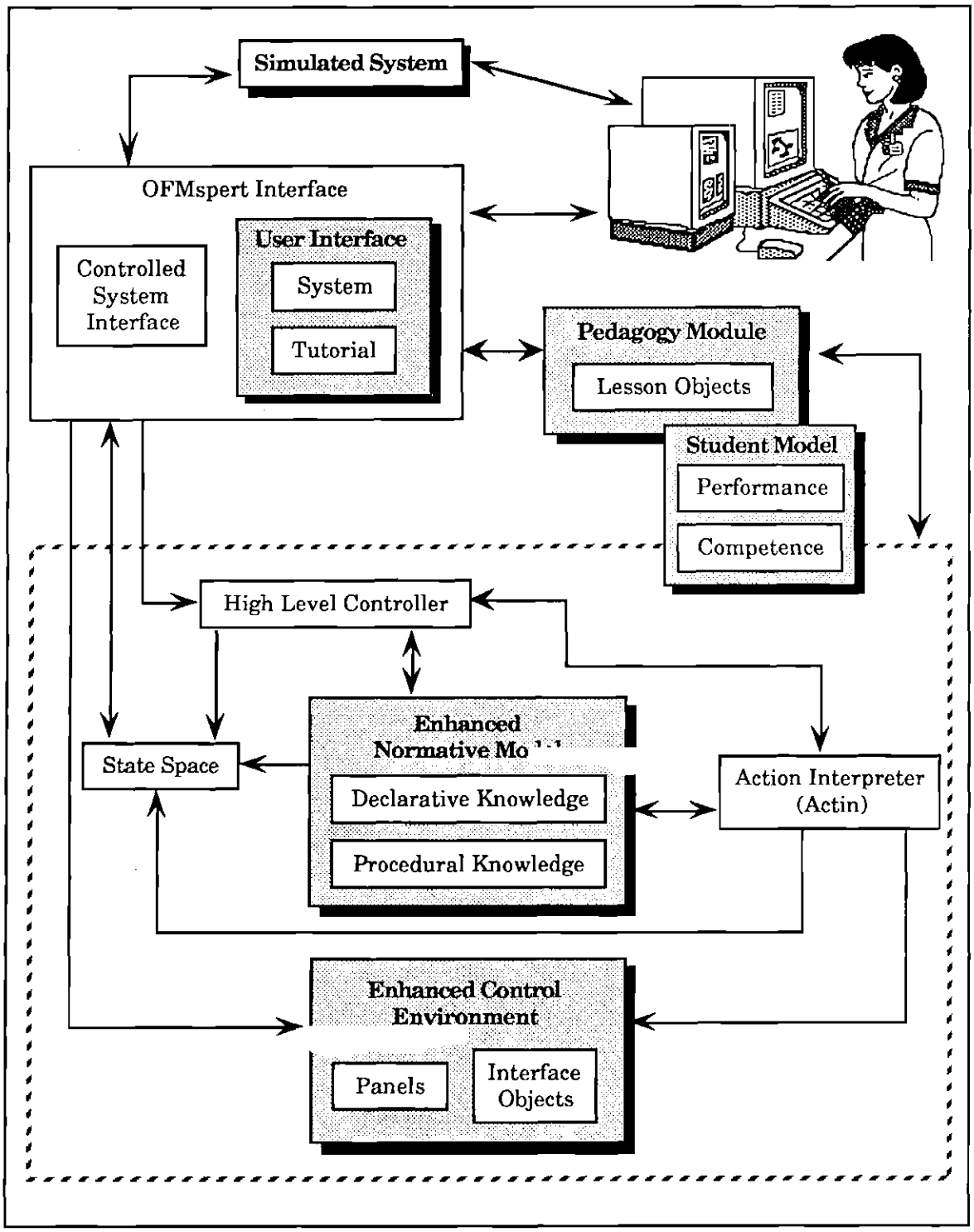


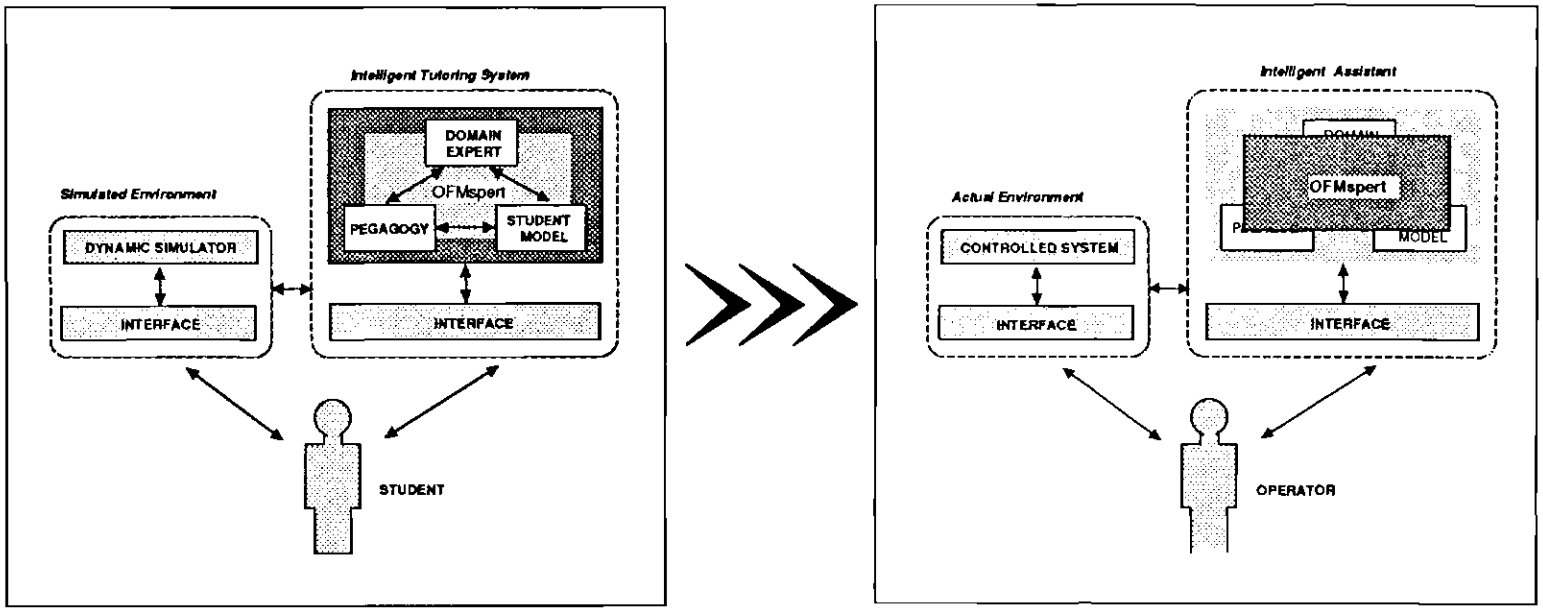
Figure 5-5. Enhanced OFMspert for Intelligent Tutoring

lesson objects (Refer to Table 4.1). Normally, the High Level Controller manages all operator actions and system events. In this enhanced architecture, the Pedagogy Module manages all student interactions and has the ability to overwrite any subsequent OFMspert processing that is normally initiated from the High Level Controller. Furthermore, depending on a lesson's goal, the Pedagogy Module controls the simulated world in terms of timing and system behavior. The Pedagogy Module relies heavily on domain specific information encapsulated in other OFMspert components, such as system component definitions and interface object characteristics. The interface object characteristics, for example, enable the tutor to manipulate the graphical displays of system components and states in order to impart the appropriate knowledge specified by a lesson object.

OFMspert's Knowledge of the Student. OFMspert's student model can be collectively represented with assessments from ACTIN and from the Pedagogy Module, and competence and performance models to record student's progress. Ideally, the student model also includes a limited-bug library.

In summary, Figure 5-6 shows the architectural evolution of the tutor to an assistant. During the training process, the student interacts with the intelligent tutor within a distributed albeit simulated environment. After training, the student, now a competent operator, interacts with the intelligent assistant within the actual control environment. The architectures also depict the overlapping role of OFMspert. During training, the tutor builds on the capabilities of OFMspert to teach effectively. After training, OFMspert can be a more effective operator's assistant with enhanced knowledge of the tutor about the domain, teaching and the student.

The operator function model and the OFMspert architecture for intelligent tutoring are instanciated for the application domain of satellite ground control. A prototype intelligent tutoring system was developed for novice operators based on the enhanced OFMspert architecture. The application domain and the associated operator function model are described in the next chapter.



During Training *During Online Operations*

Figure 5-6. Architectural evolution of an intelligent tutor to an operator's assistant

CHAPTER VI

DOMAIN OF APPLICATION:
THE GEORGIA TECH PAYLOAD OPERATIONS CONTROL CENTER

The domain of application for this research project is satellite ground control at NASA Goddard Space Flight Center (GSFC). Goddard Space Flight Center is primarily responsible for the design, launch, and control of near-earth unmanned scientific satellites. Each spacecraft has a dedicated Mission Operations Room within which personnel configure and monitor communication links to the spacecraft, monitor telemetry data transmitted from the spacecraft, and uplink spacecraft commands.

This chapter first discusses the motivation for and the initial analysis of the NASA domain followed by a general description the NASA domain. Next, the simulated NASA environment for this research is described.

Motivation and Initial Analysis of NASA Domain

This research can be viewed as a direct response to the training needs and problems at Mission Operations Division of GSFC. Current training methods include traditional on-the-job training, quizzes, and pre-launch data simulation. Besides addressing training issues for existing mission control systems, this project provides the testing grounds for ideas and concepts in training and aiding for the design of future mission control systems.

Consequently, a necessary first step of the project is to capture the NASA task environment in a real-time simulation that serves as the testbed for validating ongoing research. To do so, a detailed analysis of the NASA data/information system was conducted. The focus of the analysis was also on the operator activities within the Mission Operations Room for two existing satellites: the Earth Radiation Budget

Satellite (ERBS) and the Cosmic Background Explorer (COBE). From the analysis, general knowledge and prototypical task requirements for a satellite ground controller are identified.

Besides constraining the simulated NASA system to be developed, the knowledge and task requirements identified defined the overall instructional goals of the prototypical intelligent tutoring system to be implemented. That is, from a training point of view, the analysis conducted represent the need assessment phase in the development of any training program (Goldstein, 1986). For developing any intelligent systems, the analysis can be viewed as the knowledge engineering or acquisition phase in which knowledge to be represented is defined (Johnson, 1988). The detailed analysis was inherently an iterative process, and involved extensive observation of operator activities in real time, off-line questioning of expert operators on their activities, and generous assistance from training personnel who were experienced operators themselves. The outcome of this initial needs assessment and knowledge acquisition phase is documented in Jones, Chu, and Mitchell, 1990a; Chu and Jones, 1990a, 1990b.

The NASA Data/Information System

Figure 6-1 is a simple illustration of the NASA data/information system for the monitoring and control of near-earth scientific satellites. A spacecraft (satellite) carries one or more scientific instruments used to collect data about some phenomena; e.g., COBE's three instruments measure background radiation presumably left over from the "Big Bang". Periodically, the spacecraft is scheduled to transmit telemetry data to the ground.

Telemetry data consist of scientific data collected from the instruments and of spacecraft health and safety data. Data may be collected in real time or played back from one of the spacecraft's onboard tape recorders.

Telemetry data are either transmitted through the Tracking and Data Relay Satellite System (TDRSS) to the White Sands Ground Terminal/NASA Ground Terminal (Path A) or directly to a ground network facility (Path B). In either case, data are next transmitted to the Multisatellite Operations Control Center

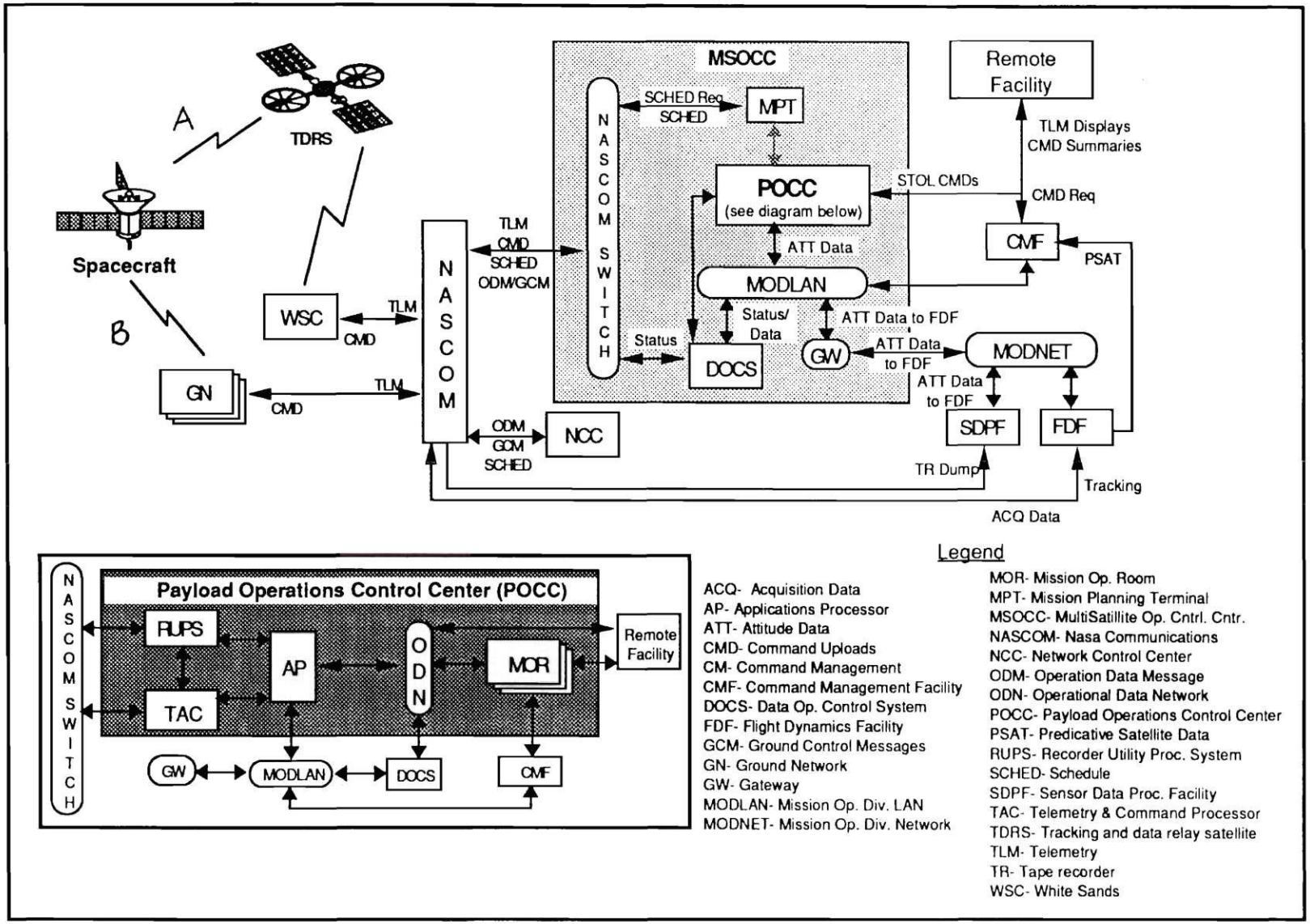


Figure 6-1. The NASA data/information system

(MSOCC) at NASA Goddard Space Flight Center in Greenbelt, MD. Scientific data are also sent to the Telemetry Processing Facility, and acquisition data are sent to the Flight Dynamics Facility.

The Network Control Center schedules network resource allocation and monitors and analyzes network performance. It communicates with MSOCC via NASA communication lines. During a spacecraft contact ("pass"), the Network Control Center sends operation data messages (ODMs) to MSOCC to verify the communication link between the TDRS and White Sands/NASA Ground Terminal.

MSOCC houses a number of computer and communication resources shared by multiple satellites. The Payload Operations Control Center forms the heart of MSOCC. As a whole, the POCC performs three major functions: telemetry processing, command management, and interfacing with other facilities (Bailin et al., 1988). The POCC consists of a number of sharable computer resources: Telemetry and Command computers (TACs), Applications Processors (APs), and Recorder/Utility Processor Systems (RUPSs). Also, the POCC contains a number of spacecraft-specific control rooms known as Mission Operations Rooms (MORs). The Telemetry and Command computer serves as a front end between NASA communications and the Applications Processor. The Recorder/Utility Processor System stores data to tape. The Applications Processor contains spacecraft-specific software to monitor and control the spacecraft, process telemetry data, display data in the Mission Operations Room, log all major system events, and communicate with support facilities external to MSOCC. Each Mission Operations Room is staffed by a Flight Operations Team (FOT) 24 hours a day, seven days a week. The Flight Operations Team analysts are responsible primarily for monitoring the health and safety of the spacecraft during its real-time supports. The analysts also monitor the communication links between the spacecraft and the ground, retrieve telemetry data collected on the spacecraft's tape recorders, upload commands into the spacecraft's command storage memory, and perform various data analysis tasks. From an operational perspective, there are three types of real-time supports: normal, routine monitoring; tape recorder playback events, where the analysts command the spacecraft's tape recorders to play back scientific data to the ground, and command storage memory load events, in which the analysts uplink and validate a file containing the next day's normal memory commands.

The GT-POCC Simulation

The Georgia Tech Payload Operations Control Center (GT-POCC) is an interactive, real-time simulation of the NASA data-information system for the capture and processing of data from near-earth scientific unmanned satellites. It provides a testbed within which new technology for satellite ground control operators can be tested and evaluated with a high level of fidelity and without the risks associated with changing the real environment. This section provides a brief overview of the simulation components and behavior of the NASA system. More information can be found in Chu, Jones, and Mitchell (1991).

GT-POCC represents three major areas of functionality: spacecraft, ground, and data/communications. The following discussion examines each of these.

Spacecraft

GT-POCC represents one hypothetical spacecraft named GASP (Goddard Atmospheric Space Platform). Figure 6-2 illustrates the composition of a *Spacecraft*, which is composed of a number of *Science Instrument* objects and four subsystems: *Radio Frequency Subsystem*, *Command and Data Handling Subsystem*, *Power Subsystem*, and *Attitude Control Subsystem*. GASP contains two science instruments named ERBE and SAGE and one subsystem of each type.

The Radio Frequency Subsystem is responsible for radio frequency communications between the spacecraft and the ground. It is composed of *Antennas* and *Transponders*. Transponders are composed of an oscillator, transmitter, and receiver. GASP's radio frequency subsystem contains two antennas and two transponders.

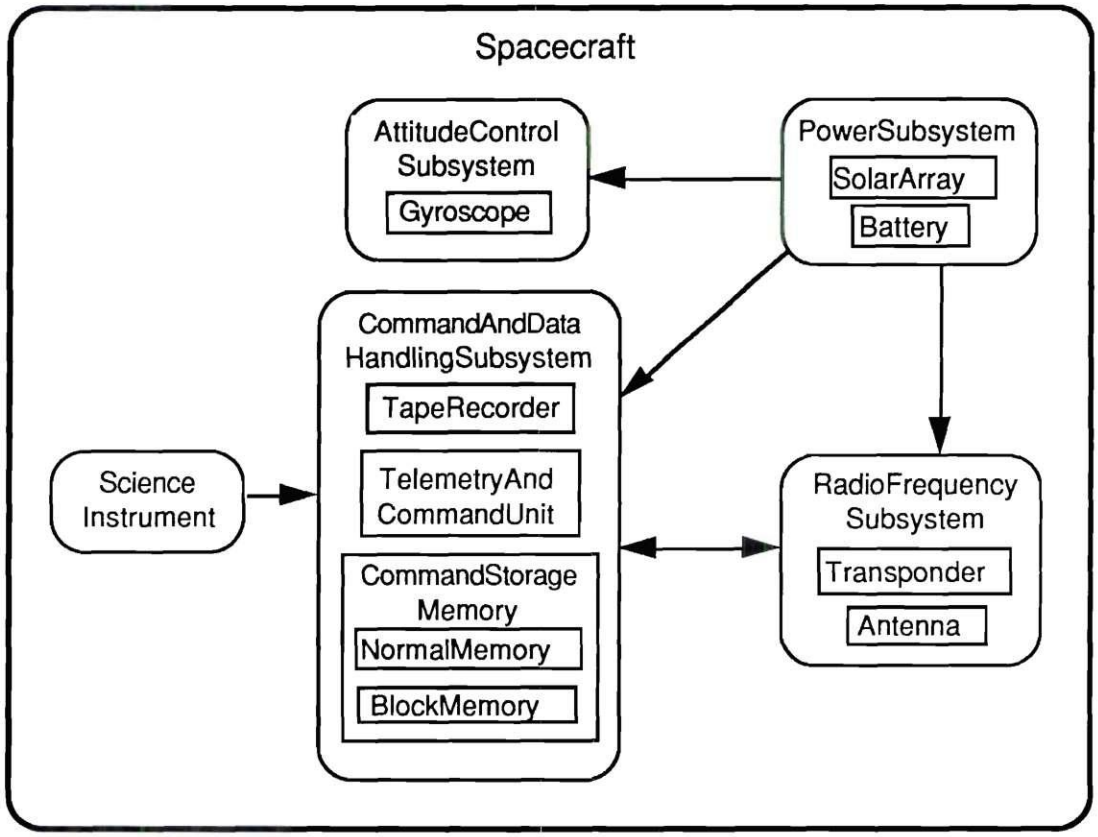


Figure 6-2. The composition of the spacecraft object.

The Command and Data Handling Subsystem manages the execution of spacecraft commands in the command storage memories and the collection and temporary storage of scientific data. It contains objects of type *Command Storage Memory*, *Tape Recorder*, and *Telemetry And Command Unit*. GASP's command and data handling subsystem contains two command storage memories (CSM1 is primary, CSM2 is backup), two tape recorders to record data from science instruments (in particular, TR1 records ERBE data and TR2 records SAGE data), and two telemetry and command units that process and send scientific data to the tape recorders (TCU1 processes ERBE data and sends them to TR1; TCU2 processes SAGE data and sends them to TR2).

The Power Subsystem supplies electrical power derived from solar arrays and batteries to the rest of the spacecraft. It contains objects of type *Battery*, *Bus*, and *Solar Array*. GASP's power subsystem contains two batteries, two solar arrays, and two buses, the Essential Bus and the Non-Essential Bus.

The Attitude Control Subsystem determines and controls the spacecraft's orientation. It contains *Gyroscope* objects. GASP contains three gyroscopes (the x, y, and z gyroscopes for the three axes of rotation).

Spacecraft components are characterized by a number of parameters that define the state of the spacecraft. These parameters may be continuous or discrete. Continuous parameters have real number values (e.g., a battery's electrical charge). Discrete parameters represent a component's configuration as a character string (e.g., a tape recorder's mode is "fast forward", "playback", "standby", or "record"). The complete set of GT-POCC spacecraft parameters is shown in Table 6-1.

Ground Support

GT-POCC also represents resources and facilities that exist on the ground to support the capture and processing of spacecraft data, including the White Sands/NASA Ground Terminal facility, three NASA communication lines, the Science Data Processing Facility, the Network Control Center, and the Multisatellite Operations Control Center (MSOCC). GT-POCC's representation of MSOCC includes three Recorder Utility Processor Systems, three Application Processors, and three Telemetry and Command Computers. Figure 6-3 shows the configuration of MSOCC for the GT-POCC system.

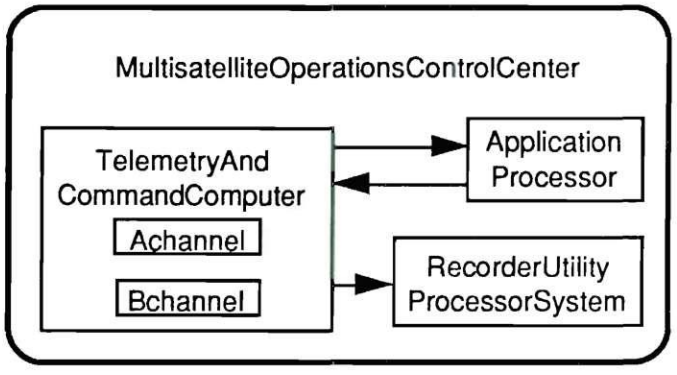


Figure 6-3. Relationship between MSOCC components in GT-POCC

Table 6-1. GT-POCC spacecraft parameters

Parameter Name	Type	Description
BAT1 charge	continuous	Battery 1 charge
BAT2 charge	continuous	Battery 2 charge
BAT1 cdRatio	continuous	Battery 1 charge-to-discharge ratio
BAT2 cdRatio	continuous	Battery 2 charge-to-discharge ratio
ESB current	continuous	Essential Bus current
ESB voltage	continuous	Essential Bus voltage
NEB current	continuous	Non-Essential Bus current
NEB voltage	continuous	Non-Essential Bus voltage
XP1 AGC	continuous	Transponder 1 signal strength
XP2 AGC	continuous	Transponder 2 signal strength
NT1 angle	continuous	Antenna 1 angle
NT2 angle	continuous	Antenna 2 angle
GYROX angle	continuous	X-axis gyroscope angle
GYROY angle	continuous	Y-axis gyroscope angle
GYROZ angle	continuous	Z-axis gyroscope angle
NT1 usage	discrete	Antenna 1 station (null, TDE, TDW, etc.)
NT1 fwd lock	discrete	Antenna 1 forward lock (yes or no)
NT1 rtn lock	discrete	Antenna 1 return lock (yes or no)
NT2 usage	discrete	Antenna 2 station (null, TDE, TDW, etc.)
NT2 fwd lock	discrete	Antenna 2 forward lock (yes or no)
NT2 rtn lock	discrete	Antenna 2 return lock (yes or no)
XP1 power	discrete	Transponder 1 power (on or off)
XP1 mode	discrete	Transponder 1 mode (coherent or non-coherent)

Table 6-1. GT-POCC spacecraft parameters (Cont'd)

Parameter Name	Type	Description
XP1 station	discrete	Transponder 1 station (null, TDE, etc.)
XP2 power	discrete	Transponder 2 power (on or off)
XP2 mode	discrete	Transponder 2 mode (coherent or non-coherent)
XP2 station	discrete	Transponder 2 station (null, TDE, etc.)
TR1 power	discrete	Tape Recorder 1 power (on or off)
TR1 mode	discrete	Tape Recorder 1 mode (standby, record, fast forward, playback)
TR1 rate	discrete	Tape Recorder 1 playback rate (high or low)
TR1 position	discrete	Tape Recorder 1 position of tape (top, middle, bottom)
TR1 format	discrete	Tape Recorder 1 data format (science)
TR1 channel	discrete	TAC channel which carries Tape Recorder 1 data (the q channel)
TR2 power	discrete	Tape Recorder 2 power (on or off)
TR2 mode	discrete	Tape Recorder 2 mode (standby, record, fast forward, playback)
TR2 format	discrete	Tape Recorder 2 data format (science)
TR2 channel	discrete	TAC channel which carries Tape Recorder 2 data (the q channel)
CSM1 mode	discrete	Command Storage Memory 1 mode (enabled, disabled, load, dump)
CSM1 section	discrete	Command Storage Memory 1 partition of memory that is currently executing (normal or block)
CSM2 mode	discrete	Command Storage Memory 2 mode (CSM2 is always disabled)
CSM2 section	discrete	Command Storage Memory 2 partition of memory that is currently executing (CSM2 is always "none")

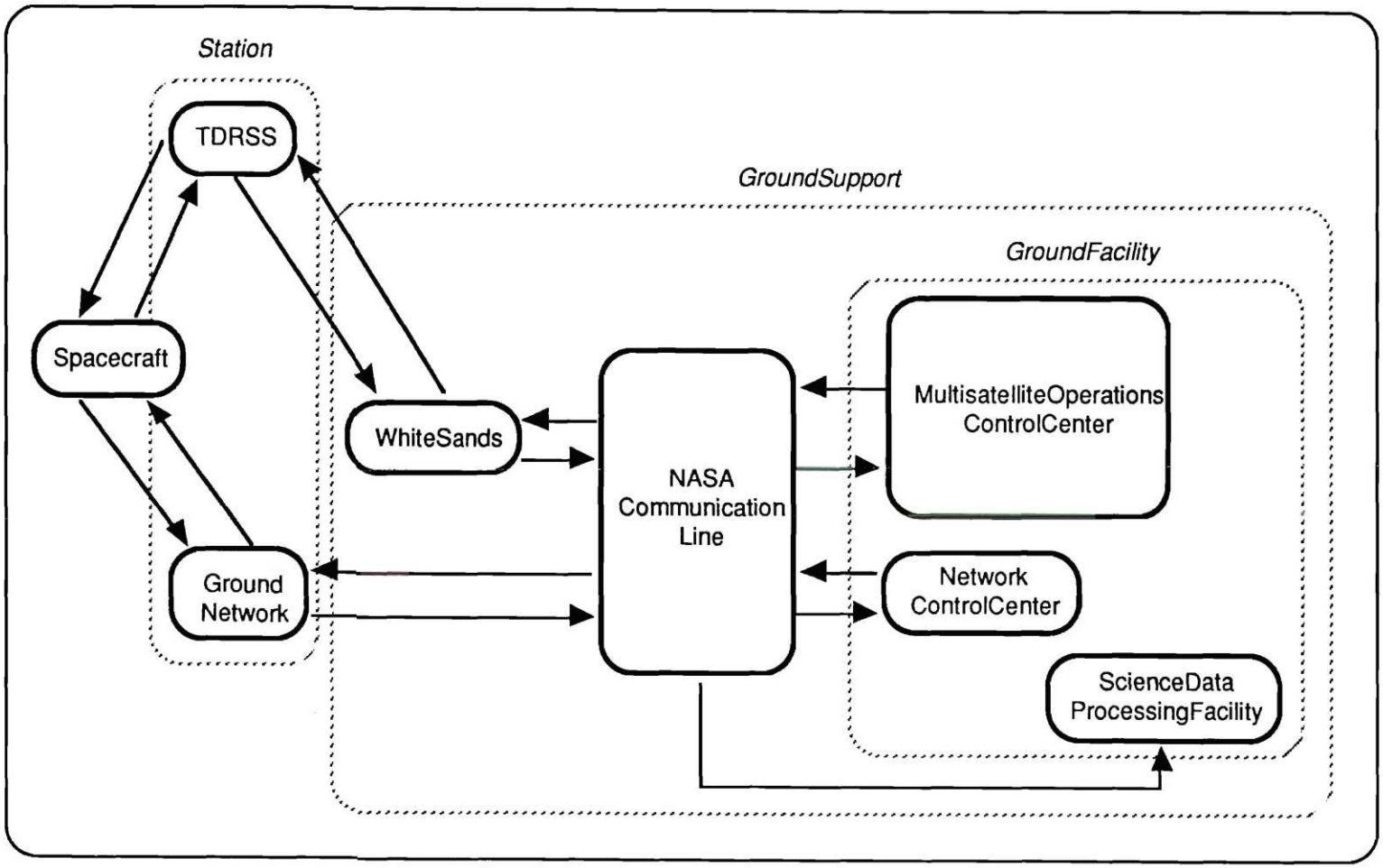


Figure 6-4. Major GT-POCC components for the NASA data/information system

Data/Communications

Real-time supports may utilize TDRSS or ground network resources. GT-POCC supports two TDRS satellites (TDRS East and TDRS West) and three ground network facilities (Madrid, Goldstone, and Wallops Flight Facility). Figure 6-4 summaries the major GT-POCC components with respect to the NASA data-information network.

GT-POCC Operator Interface

The GT-POCC simulation itself is designed to run with any number of client processes that may support operator interaction. This section describes the implementation of the operator interface that mimics the current controls and displays of an actual Mission Operations Room.

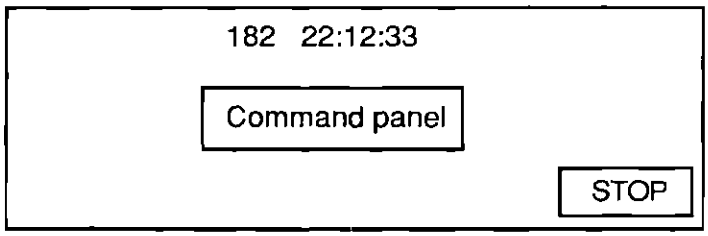


Figure 6-5. The initial GT-POCC user interface window.

The GT-POCC operator interface consists of multiple, overlapping windows on a single display screen that supports mouse and keyboard interaction. When the GT-POCC program begins, a small initial window appears (see Figure 6-5) that shows the simulation time. This window contains a button labeled "Command Panel". Clicking this button causes the command panel to appear (see Figure 6-6). The command panel is the central means of interaction with GT-POCC. The top part of the command panel is labeled "Command Panel" and contains buttons to run STOL (System Test and Operation Language) procedures (e.g., the APSET button executes a procedure to set up the Application Processor). The bottom part of the command panel is labeled "Display Pages" and has buttons to display other windows.

Command Panel Procedures. Command panel procedures are executed by clicking the appropriate button(s). This will cause the button(s) to turn yellow, which indicates that the associated

procedure is pending. Procedures will not be executed until the START button is clicked. When the START button is clicked, any yellow buttons turn green, as does the START button, and the procedures begin execution. The chosen buttons and the START button will turn back to white. Pending procedures can be cancelled by clicking again on a yellow button or clicking the CANCEL button.

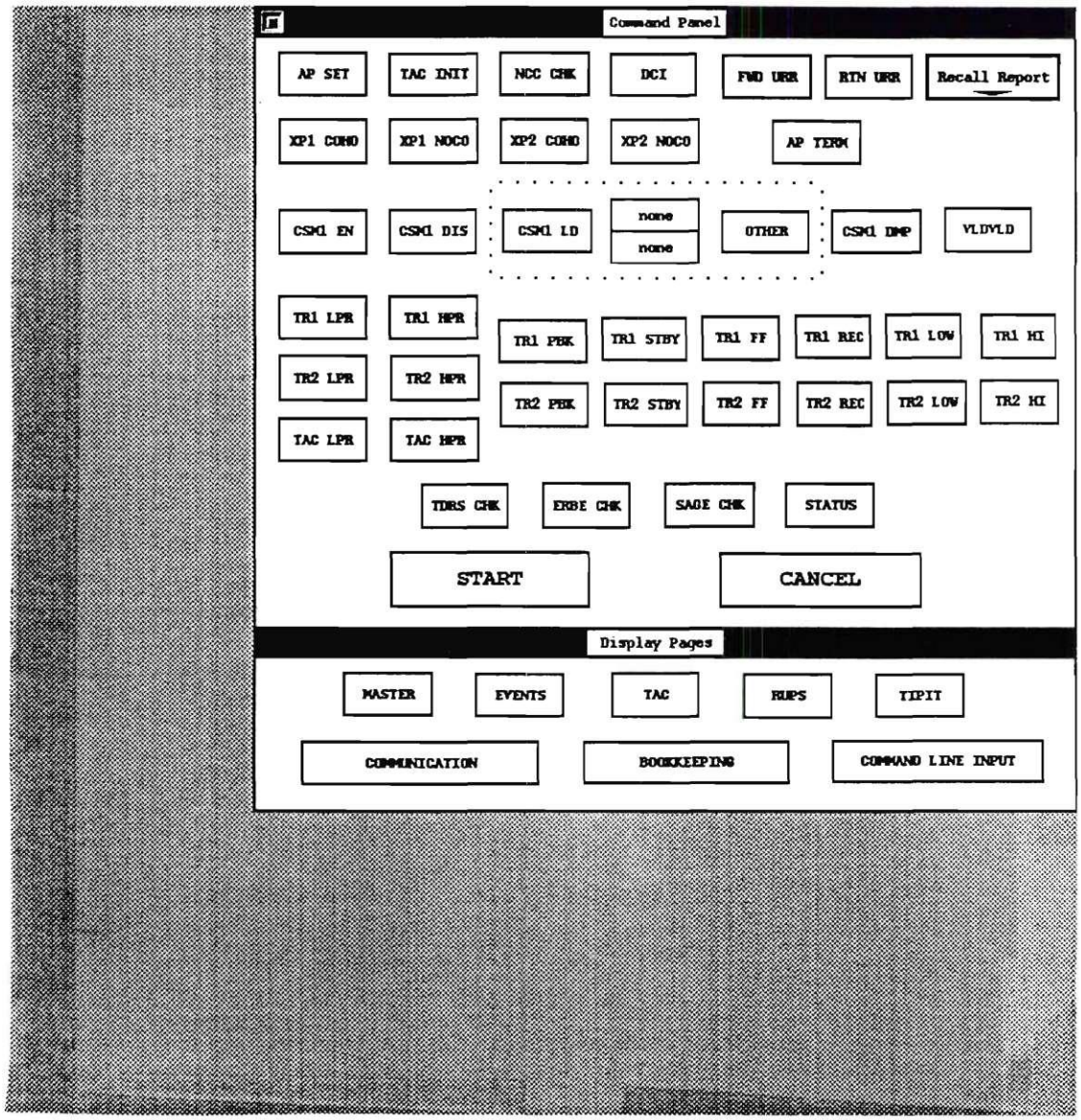


Figure 6-6. The GT-POCC command panel

Unlike the actual MOR environment, the GT-POCC command panel is not reconfigurable; i.e., the button names remain constant. Also, the GT-POCC command panel is context-sensitive with respect to types of support events; buttons associated with commanding the spacecraft's tape recorders or command storage memory are only active for the appropriate support events.

The command panel buttons are organized roughly in time order. The eight top left buttons (i.e., AP SET, TAC INIT, NCC CHEK, DCI, XP1 COHO, XP1 NOCO, XP2 COHO, XP2 NOCO) refer to procedures that are to be done before or at the beginning of the support. AP SET is the procedure to set up the Applications Processor. TAC INIT is the procedure to initialize the Telemetry and Command Computer. NCC CHEK is the procedure to initiate communications with the Network Control Center. This requests the NCC to begin sending periodic Operation Data Messages (ODMs) to the POCC. DCI is the procedure to perform Doppler compensation inhibition, which is necessary for coherent supports. The remaining four buttons reset the mode of the indicated transponder; i.e., XP1 COHO sets the mode of transponder 1 to coherent, XP1 NOCO sets the mode of transponder 1 to non-coherent, and XP2 COHO and XP2 NOCO do the same for transponder 2. Resetting the transponder mode is only necessary if the transponder was misconfigured.

The upper right portion of the command panel contains three buttons: FWD URR, RTN URR, and APTERM. FWD URR refers to a forward link operator reacquisition request, and is performed when the forward link signal (from ground to spacecraft) is lost. RTN URR refers to a return link operator reacquisition request, and is performed when the return link signal (from spacecraft to ground) is lost. APTERM is the procedure to terminate the Applications Processor, and needs to be done at the end of every support.

The next line of buttons on the command panel is related to command storage memory load events. CSM1 EN sets the mode of Command Storage Memory 1 (CSM1) to "enabled". CSM1 DIS sets the mode of CSM1 to "disabled". CSM1 LD sets the mode of CSM1 to "load". The next buttons are the names of CSM load files. CSM1 DMP is the procedure to dump the contents of CSM1's normal memory to the ground. VLDVLD is the procedure to validate this "dump data" against the original CSM load on the ground.

The next group of buttons is related to tape recorder playback events. The six leftmost buttons are procedures to set the playback rate to low and high (LPR = low playback rate; HPR = high playback rate) for tape recorder 1 (TR1), tape recorder two (TR2), and the TAC for this support. The next buttons correspond to procedures to command a tape recorder (TR1 or TR2) to playback (PBK), standby (STBY), fast forward (FF), record (REC), to record at a low rate (LOW), and to record at a high rate (HI).

The last row of buttons is related to monitoring data. TDRS CHK is a procedure to check on TDRSS parameters. ERBE CHK is a procedure to monitor parameters from the ERBE science instrument. SAGE CHK is a procedure to monitor parameters from the SAGE science instrument. The STATUS procedure brings up a series of display pages that show spacecraft health and safety data. These pages are redundant with the MASTER page described below.

Display Pages. Other display pages can be accessed by clicking on the appropriate button. The MASTER page (see Figure 6-7) shows the status of all of GASP's parameters. Parameters are either discrete (e.g., a tape recorder's power is "on" or "off") or continuous, real-valued numbers (e.g., the charge of a battery). Each real parameter has specified ranges of values that are normal, marginal low, marginal high, out of limits low, and out of limits high. The display of a current value for a real parameter is color coded with respect to these limits: values within the normal range are colored green, values in the marginal ranges are colored yellow, and values in the out of limits ranges are colored red.

The parameters shown on the MASTER page are as follows. For each antenna, the station to which it is transmitting (a TDRS, a ground network facility, or NULL), forward and return link status ("Yes" if signal acquired, "No" if not), and the value of its angle is shown. For each transponder, its power, mode, station, and value of its signal strength (AGC) are shown. For each battery, the values of its charge and charge-to-discharge ratio are shown. For each electrical bus, the values of its voltage and current are shown. For each gyroscope, the value of its angle is shown. For each tape recorder, its power, mode, rate, position, format, and channel of transmission are shown. For each command storage memory, its mode and current section are shown.

The EVENTS page (see Figure 6-8) provides a listing of relevant messages. These include echoes of operator actions, confirmation that procedures were executed appropriately, and alert messages from other facilities.

The COMMUNICATION page (see Figure 6-9) is the GT-POCC version of voice-link communications. To verify communications with and send alert messages to NCC, the DOCS, TNC, White Sands, and TPF, the operator clicks on the appropriate message(s) and then clicks the "Okay" button.

The TAC page (see Figure 6-10) gives detailed information on the state of the current TAC in terms of cumulative data counts and error counts at each channel. When data are being processed through the TAC during a real-time support, the data counts are colored in

green. Should the polynomial error counts or synchronization loss counts exceed a specified number, those counts are colored in red to denote a possible failure.

The RUPS page (see Figure 6-11) shows the start and stop times and total block count of tape recorder playback data received at the RUPS. The TIPIT page (see Figure 6-12) shows the start and stop times and total block counts of tape recorder playback data received at the Telemetry Processing Facility (TPF).

The BOOKKEEPING page (see Figure 6-13) is the GT-POCC mechanism for filling out pass plan sheets and reports. This page shows buttons that bring up further display pages. The PASS PLAN page (see Figure 6-14) is partially filled in with information about the current support; a pass plan must be completed for every support. The DATA ACCOUNTABILITY page (see Figure 6-15) is to be filled out for tape recorder playback events. For any communication link or MSOCC equipment problems, an EVENT REPORT (see Figure 6-16) is required. Similarly, for any spacecraft or command storage memory anomaly (i.e., a real parameter that is out of limits or a command storage memory location with anomalous data), the ANOMALY REPORT button causes a menu to appear from which Spacecraft Anomaly or CSM Anomaly report can be selected. Figure 6-17 shows the Spacecraft Parameter Anomaly Report. Figure 6-18 shows a CSM Anomaly Report.

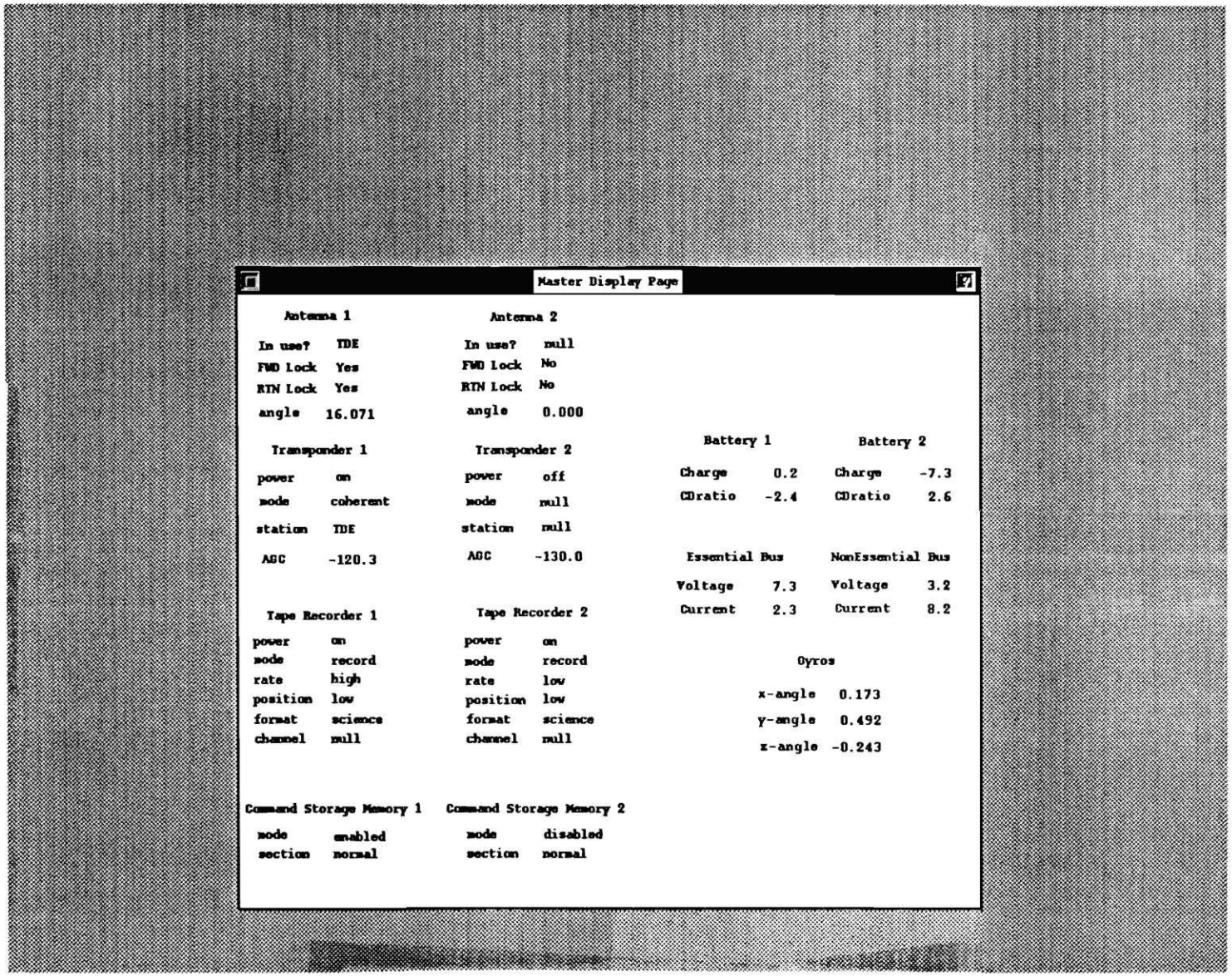


Figure 6-7. The GT-POCC MASTER display page.

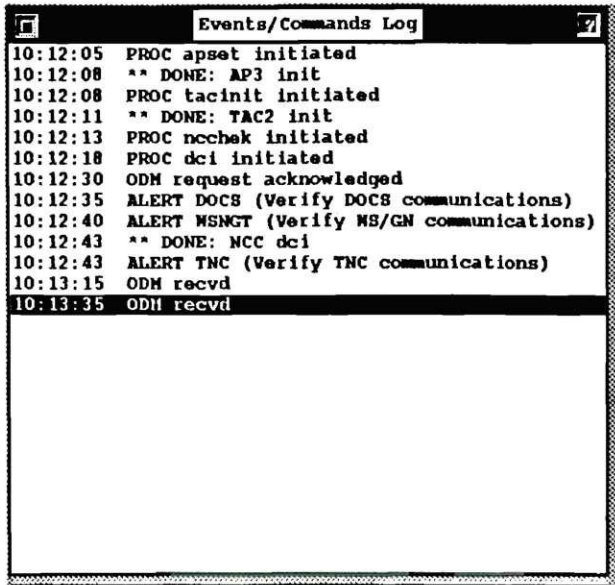


Figure 6-8. The GT-POCC EVENTS display page.

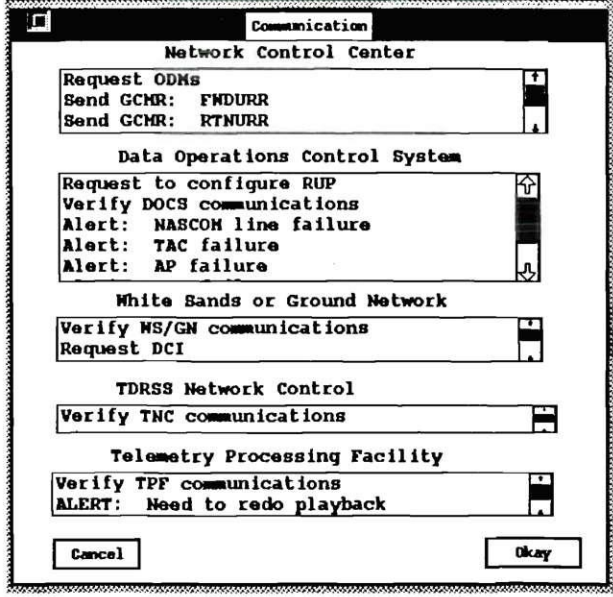


Figure 6-9. The GT-POCC COMMUNICATION display page.

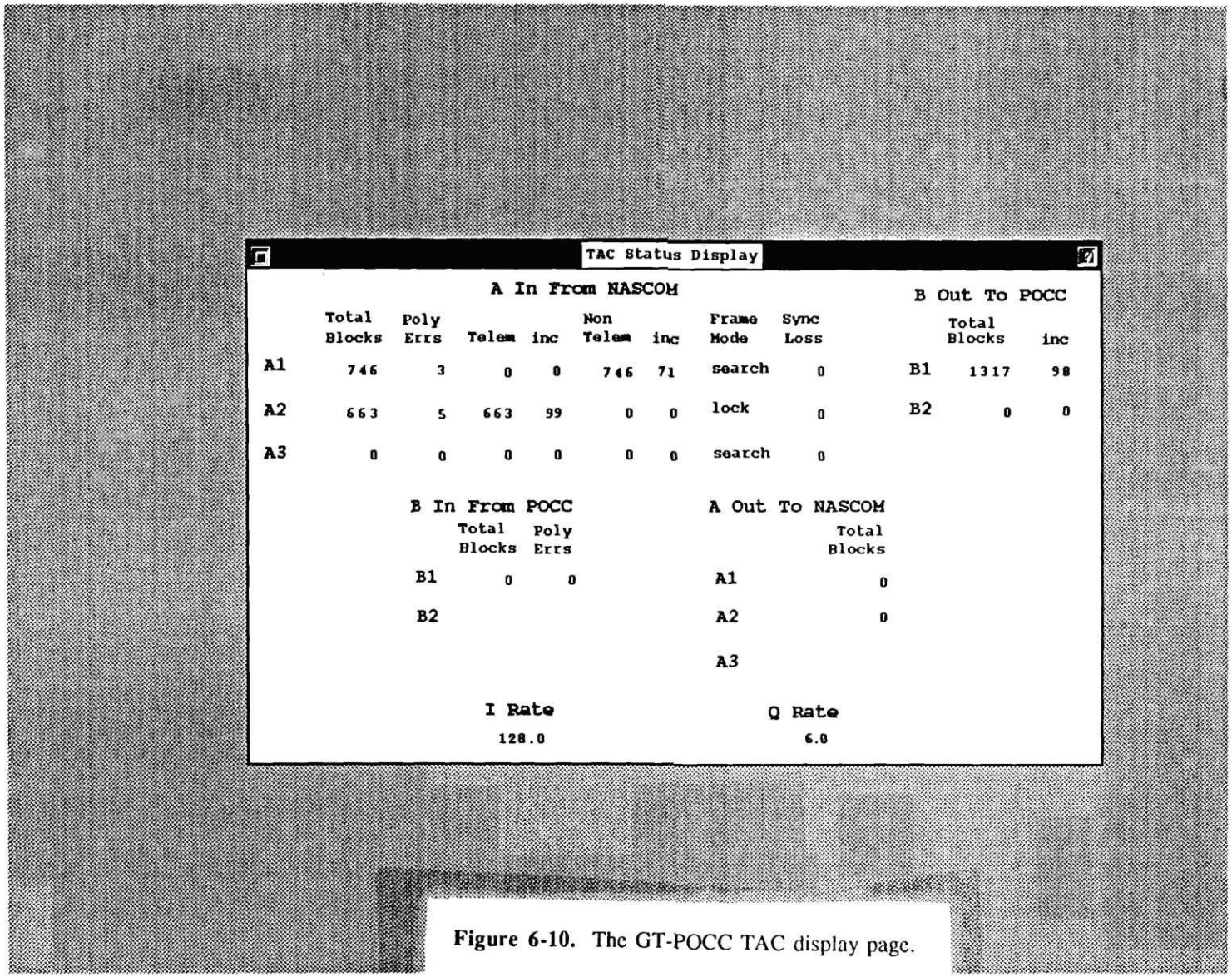


Figure 6-10. The GT-POCC TAC display page.

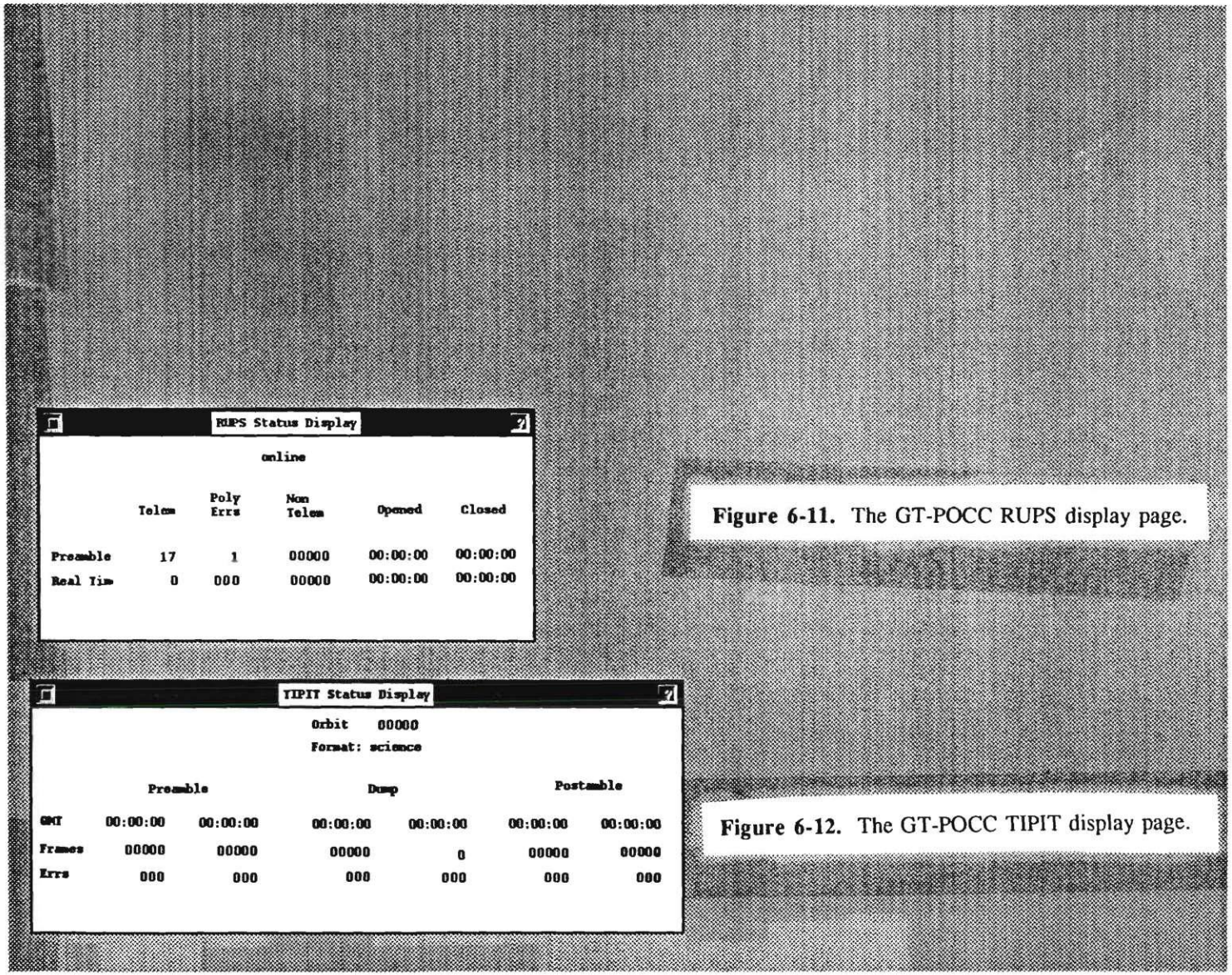


Figure 6-11. The GT-POCC RUPS display page.

Figure 6-12. The GT-POCC TIPIT display page.

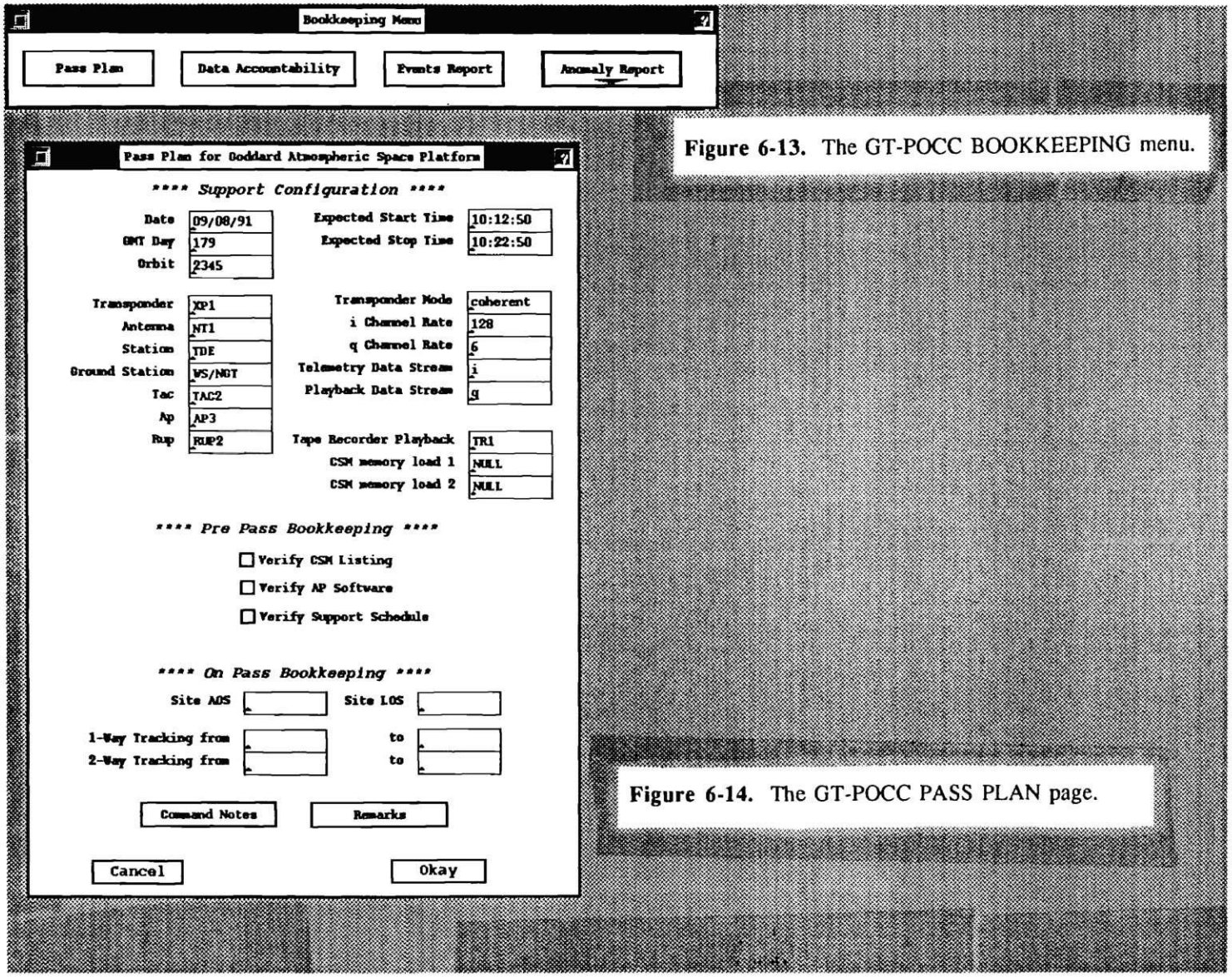


Figure 6-13. The GT-POCC BOOKKEEPING menu.

Figure 6-14. The GT-POCC PASS PLAN page.

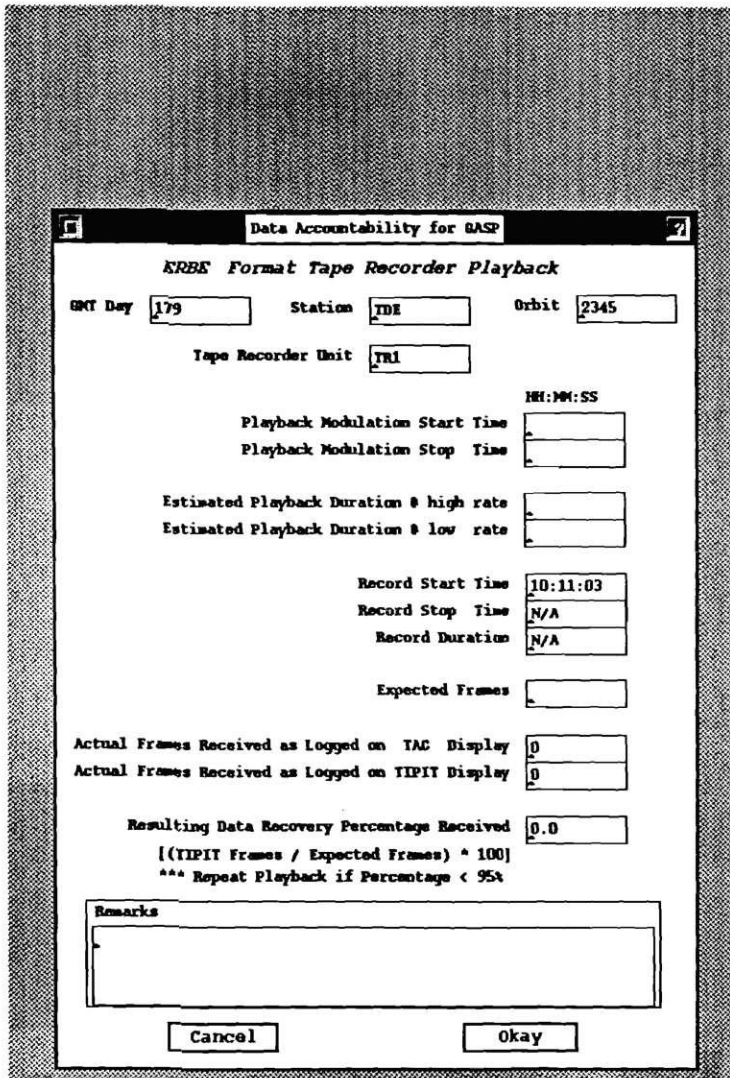


Figure 6-15. The GT-POCC DATA ACCOUNTABILITY.

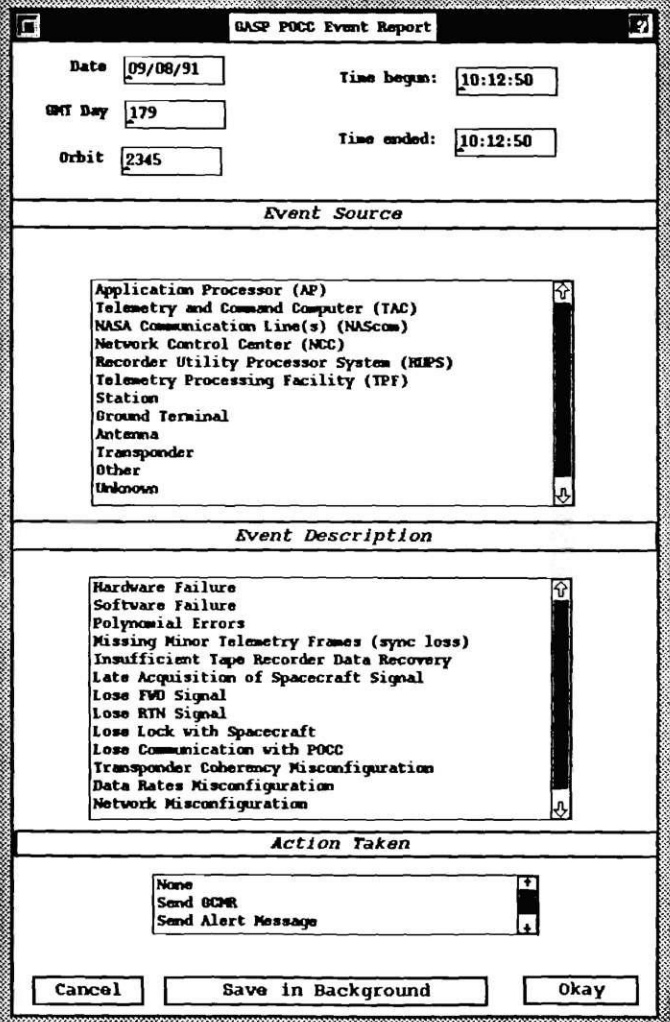


Figure 6-16. The GT-POCC EVENT REPORT page.

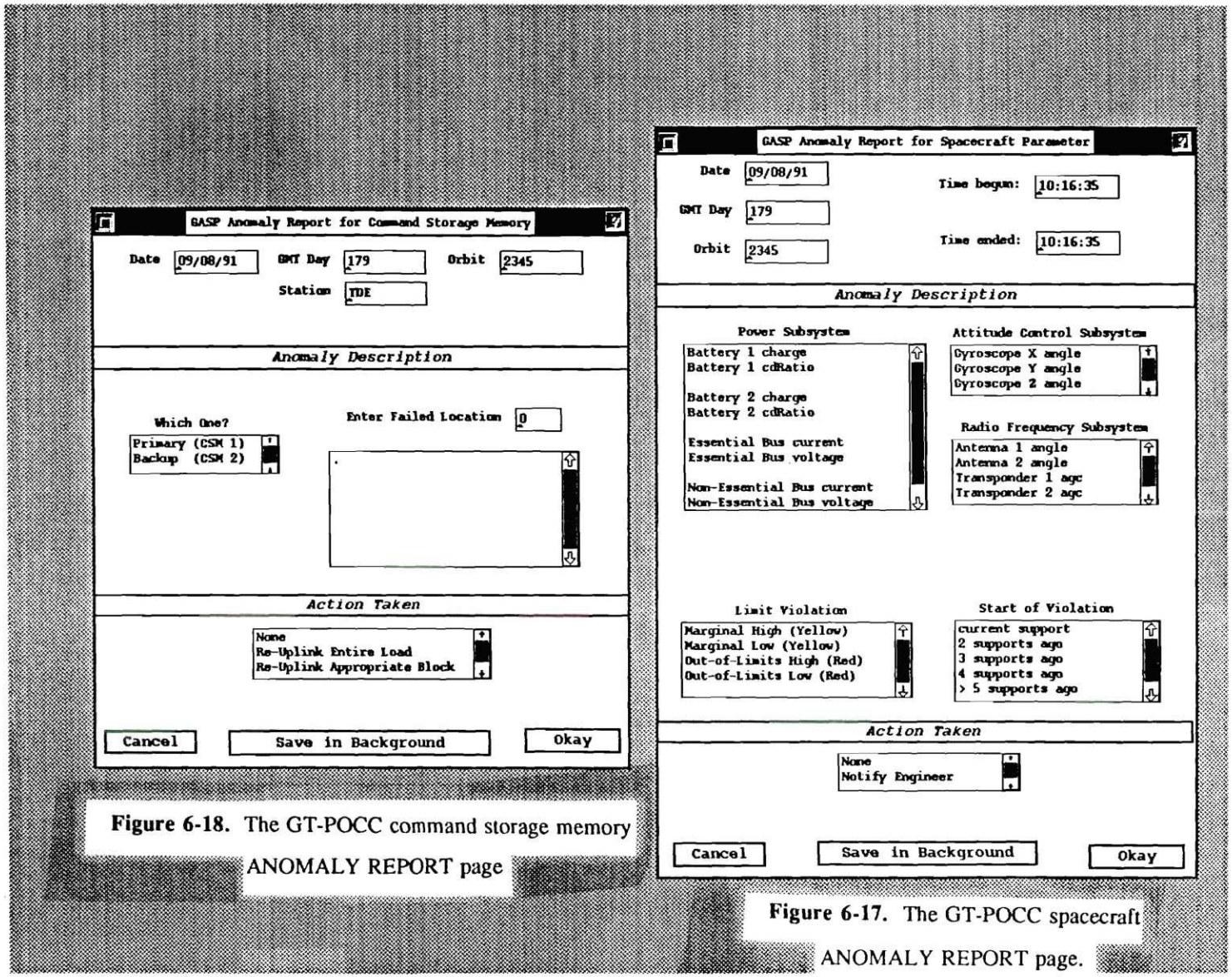


Figure 6-18. The GT-POCC command storage memory ANOMALY REPORT page

Figure 6-17. The GT-POCC spacecraft ANOMALY REPORT page.

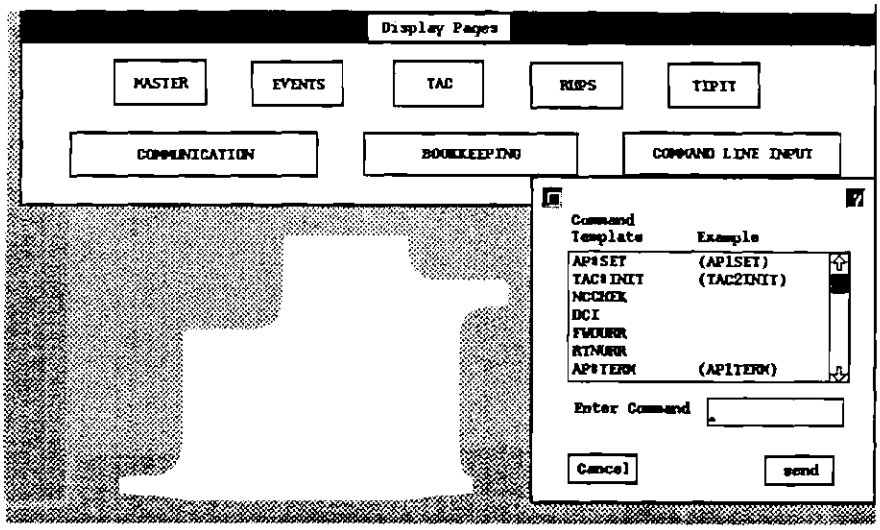


Figure 6-19. The GT-POCC command line input display

The COMMAND LINE INPUT page (see Figure 6-19) allows for the manual typing of STOL commands. This is sometimes necessary to override the constraints of the context-sensitive command panel.

GT-POCC Event Management

GT-POCC maintains a global list of events to execute. Events are created and added to the event list dynamically. Simulation events include the execution of commands in command storage memory, spacecraft parameter and communication link updates, and communications with clients.

Figure 6-20 illustrates the main loop of the GT-POCC simulation. After initialization, the event list is checked for the time the next event is due to be executed. If that time is the same as the current simulation time, the simulation clock is updated and that event is processed. If it is not yet time to execute the next event, the number of clients currently communicating with the simulation is checked. If this number is zero, the simulation "sleeps" until the time of the next event. If the number of clients is positive, and a message has been received from a client, that message is interpreted. This cycle is repeated until the simulation is terminated.

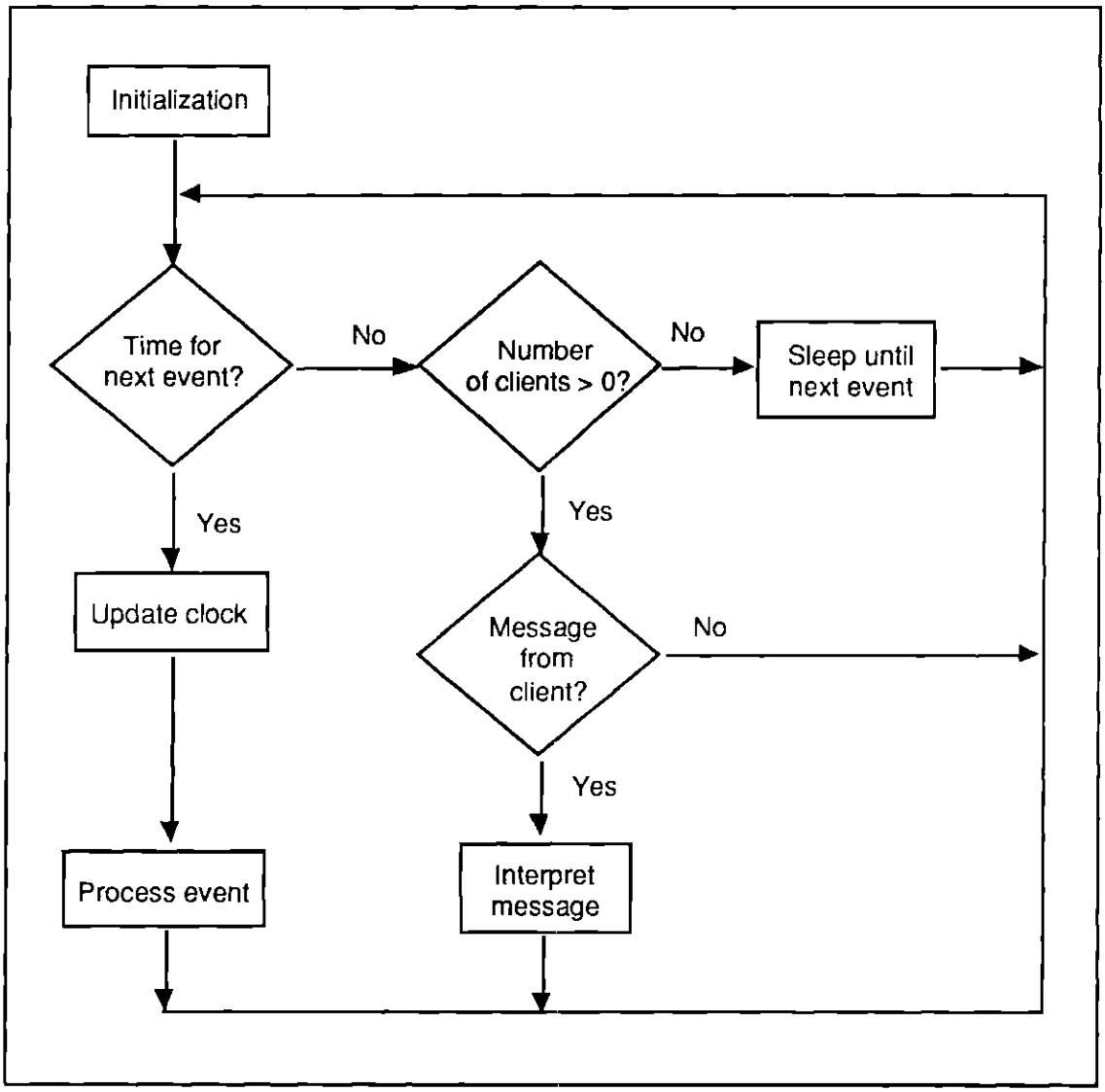


Figure 6-20. The main loop of the GT-POCC simulation.

Significant simulation events, such as the start and end of real-time supports and system failures (e.g., communication link failures and out of limits spacecraft parameters), are scheduled in a file-driven manner. The GT-POCC program is run with an argument denoting a session number. From this session number, GT-POCC reads a session file that specifies an initial GMT time and the relative start times and associated files names of the scenarios (i.e., real-time supports) that make up this session. The scenario start times are relative to the simulation start time; by convention, every session begins at simulation time zero. Figure 6-21 shows an example session and scenario file. Session X involves three real-time

supports: scenario X1 begins at time 10, scenario X2 begins at time 400, and scenario X3 begins at time 1500. Each scenario file then specifies the configuration of equipment required for the support, whether or not this support is a tape recorder playback or command storage memory load event, and the relative start times of system failures. For example, the scenario X1 file specifies that this support is orbit number 2233, lasts 300 seconds, and uses spacecraft antenna 1, spacecraft transponder 1, the TDRS East station, the White Sands ground terminal, NAS1, TAC2, and AP1. Also in scenario X1, the forward link is scheduled to be lost at 50 seconds after the start of the scenario (i.e., at simulation time 60) and reacquired 120 seconds later, and the spacecraft's battery 1 charge is scheduled to go out of limits high at 200 seconds after the beginning of the scenario. The advantage of this file-driven scheme with relative time stamps is the ease with which new sessions and scenarios can be created.

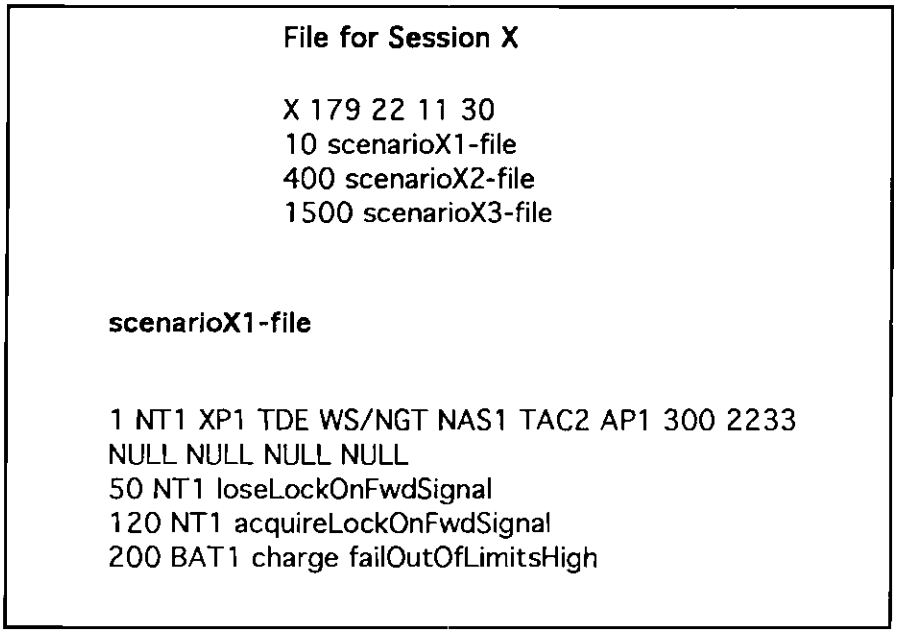


Figure 6-21. An example of a session and associated scenario file.

The GT-POCC Operator Function Model

In general, GT-POCC operator activities can be divided into three categories: pre-pass, on-pass and post-pass. Pre-pass activities include configuration of system components, verification of communications, software, and schedules, and initialization of other support-dependent parameters. On-pass activities include monitoring, spacecraft commanding, and bookkeeping. Post-pass activity consists of termination of ground equipment. Bookkeeping is also performed during pre-pass setup and post-pass termination.

The top-level heterarchic structure of these activities as an operator function model is given in Figure 6-22. The operator function model structure is extended in include a representation of phase (after Verfurth, 1991). The pre-pass phase is defined in GT-POCC as two minutes before the acquisition of telemetry data. During the pre-pass phase, the operator is responsible for managing three concurrent functions: configuration, verification, and "set up" of other parameters. Once the signal has been acquired (i.e., the phase becomes on-pass), the operator is responsible for carrying out the major functions of monitoring, commanding, and bookkeeping. After data flow has been terminated, the post-pass phase begins, and the operator is responsible for carrying out the termination function.

In the GT-POCC operator function model, the tasks level is nearly always isomorphic with the subfunctions level (e.g., the "Configure AP" subfunction decomposes into just the "Execute Configure AP" task). In such cases the tasks level description is omitted.

Pre-Pass Configuration

Figure 6-23 shows the decomposition of the configuration function. The configuration function consists of three subfunctions: to configure the Applications Processor, to configure the Telemetry and Command Computer, and, if the support is a tape recorder playback event, to configure the Recorder Utility Processor System. The Application Processor needs to be configured first before the Telemetry and Command Computer. The Recorder Utility Processor System is configured automatically during a tape recorder playback event. The operator initiates the APSET procedure via the command panel to configure

the Application Processor. Similarly, to configure the Telemetry and Command Computer, the operator initiates the TACINIT command panel procedure.

Pre-Pass Verification

Figure 6-24 shows the decomposition of the verification function. Verification consists of four concurrently-managed subfunctions: to verify communications, verify the command storage memory listing, verify the Application Processor software, and verify the support schedule for this and the next few subsequent supports. The Verify Communications subfunction is further decomposed into four concurrently-managed tasks: to verify communications with the Data Operations Control System, White Sands or ground network facility, TDRSS Network Control (if this support utilizes TDRSS), and, if this support is a tape recorder playback event, with the Telemetry Processing Facility.

The operator initiates the NCCHEK command panel procedure to accomplish the Verify with NCC task The NCCHEK procedure requests the Network Control Center to begin sending Operations Data Messages at a rate of every 20 seconds. The other tasks that constitute the Verify Communications subfunction involve the operator's action to debrief the appropriate facility. In the actual Mission Operations Room environment, the operator debriefs verbally over a voice link. In GT-POCC, the operator accomplishes debriefing electronically by clicking on the appropriate message in the Communications window.

The other three subfunctions that make up the Verification function are likewise electronically implemented. In the actual Mission Operations Room, a computer procedure is initiated to check the Application Processor software, and the command storage memory listing and support schedule are checked manually from paper copies. In GT-POCC, the electronic pass plan has three checkboxes to support each of these subfunctions. The operator clicks on the "Verify CSM Listing" checkbox to bring up an electronic listing of the most recent command storage memory validation listing (or a message that none yet exists). The operator clicks on the "Verify AP Software" checkbox to bring up another window where the operator clicks a button to start the software check. The operator clicks on the "Verify Support Schedule" checkbox to bring up an electronic listing of the supports scheduled for the current GT-POCC experimental session.

Pre-Pass Setup

Figure 6-25 illustrates the decomposition of the Setup function into four subfunctions, three of which are context-dependent. The operator accomplishes pre-pass bookkeeping by verifying the electronic pass plan and jotting down any planned activities in advance. If the current support is coherent, the operator must request that Doppler compensation be inhibited via the DCI command panel procedure. If the current support's transponder is not configured as scheduled, the operator must reconfigure it with the appropriate command panel directive (i.e., if Transponder 1 is set to be non-coherent but is scheduled to be coherent, the operator should execute the XP1COHO procedure to make Transponder 1 be coherent). If the current support's Telemetry and Command computer is misconfigured in terms of data rates, the operator should alert the Data Operations Control System to rectify the problem.

Once the spacecraft signal has been acquired, data begin to arrive at the Mission Operations Room. The operator is responsible for managing three concurrent functions during this on-pass phase: monitoring, commanding, and bookkeeping.

On-Pass Monitoring

Figure 6-26 illustrates the decomposition of the Monitoring function into three concurrently-managed subfunctions: to monitor communications with other ground facilities, to monitor telemetry data, and, if an MSOCC or communication link failure occurs, to manage that failure. The Monitor Communications subfunction is further decomposed into the tasks of monitoring for messages from the Network Control Center, the Data Operations Control System, TDRSS Network Control, and the Telemetry Processing Facility. The operator accomplishes these tasks by the repetitive action of checking the EVENTS page throughout the duration of the pass. The Monitor NCC task is also accomplished by monitoring the block counts on the A1 channel of the Telemetry and Command computer display page.

The Monitor Telemetry subfunction is composed of three concurrently-managed tasks: monitoring spacecraft health and safety data, monitoring science data, and, if this support is a tape recorder playback event, monitoring the tape recorder playback data. Monitoring the spacecraft health and safety data involves the operator checking the MASTER page and executing the STATUS and TDRSSCHK command panel

procedures. Monitoring scientific data involves the operator checking the EVENTS page and executing the ERBECHK and SAGECHK command panel procedures. Monitoring the tape recorder playback data involves the operator examining the RUPS and TIPIT display pages.

The Failure Management subfunction is instanciated for every MSOCC or communication link failure. The associated action is to send the appropriate message to the appropriate facility via the Communications window. For example, if a Telemetry and Command computer channel experiences a failure, the operator would click on the "Alert: TAC Failure" message to be sent to the Data Operations Control System.

On-Pass Commanding

Figure 6-27 shows the decomposition of the Commanding function. Spacecraft commanding is a function required only for two types of supports: tape recorder playback and command storage memory load events. If the support is a tape recorder playback event, the two subfunctions to be carried out are to command the tape recorder to play back its data and, when the playback is complete, to command it to begin recording again. The actions to accomplish the Tape Recorder Playback subfunction are to first command the tape recorder to STANDBY mode and then command it to PLAYBACK model. These commands are accomplished via the command panel. If Tape Recorder 1 is to be played back, the operator would execute the TR1STBY and then the TR1PBK command panel procedures. Similarly, for Tape Recorder 2, the operator would execute the TR2STBY and then the TR2PBK command panel procedures. Once playback is complete, the operator commands the tape recorder to resume recording with either the TR1REC or TR2REC command panel procedure.

If the support is a command storage memory load event, three serial subfunctions are performed. First, the command storage memory's normal memory partition is loaded with the new command files. After the load is complete, these newly-loaded contents are dumped back to the ground for validation with the original copy of the load files. After the memory has been dumped, the command storage memory is enabled so that it can resume execution of its commands. The CSM Load subfunction consists of three ordered actions. First, the command storage memory mode is set to DISABLE with the CSM1DIS

command panel procedure. Next, the command storage memory mode is set to LOAD with the CSM1LD command panel procedure. Finally, the command files are sent up to the spacecraft with the CLOAD command. This command is executed via the reconfigurable command panel buttons which show the names of the files to be uplinked. After the load is complete as evidenced by a message on the EVENTS page, the operator performs the CSM Dump subfunction by commanding the command storage memory to DUMP mode with the CSM1DMP command panel procedure. After the dump is complete as evidenced by an EVENTS page message, the operator commands the command storage memory to resume execution with the CSM1EN (command storage memory enable) command panel procedure.

On-Pass Bookkeeping

The Bookkeeping function is managed differently in GT-POCC than in the actual Mission Operations Room. In the real operational environment, all bookkeeping is done by hand on paper, and operators are encouraged to jot down notes during the support but to save most of the work until after the support is over. In GT-POCC, all bookkeeping is done online by filling out electronic forms, and operators may quickly fill out reports during the real-time support.

GT-POCC supports five types of bookkeeping, which are modeled in Figure 6-28 as subfunctions of the Bookkeeping function. First, the operator must fill out the pass plan for every support During on-pass phase, the operator types in the times of acquisition and of one-way and two-way tracking on the pass plan form. If the support is a tape recorder playback event, the operator must fill out a data accountability form with the playback start and stop times and the data counts from the TAC and TIPIT display pages. If the support is a command storage memory load, the operator must generate and examine a validation listing that compares the original command load files (the "ground image") with the newly-dumped contents of command storage memory. This is accomplished with the VLDVLD command panel procedure. If an MSOCC or communication link problem occurs, the operator must fill out an event report stating the start and stop times, source, and description of the problem and what action, if any, was taken to remedy the problem. If a spacecraft parameter goes out of limits or a location of command storage memory

experiences an anomaly (i.e., experiences a "memory hit"), the operator must fill out an anomaly report stating the start and stop times and description of the anomaly.

Post-Pass Termination

A final function represents the requirement to deconfigure the Application Processor at the end of the real-time support. The Termination function consists of the Terminate Application Processor subfunction and the post-pass bookkeeping subfunction. The operator is required to execute the APTERM command panel procedure as shown in Figure 6-29. The operator accomplishes the post-pass bookkeeping subfunction by completing the pass plan with final details such as the time when signal is lost and general comments about events during the pass.

In summary, the GT-POCC operator function model provides a structure for the development of the enhanced OFMspert system with intelligent tutoring capabilities. The complete GT-POCC operator function model as implemented in the Georgia Tech Visual Inspectable Tutor and Assistant (GT-VITA) system is given in Table 6-2. The full specification of functions, subfunctions, tasks, and actions is shown.

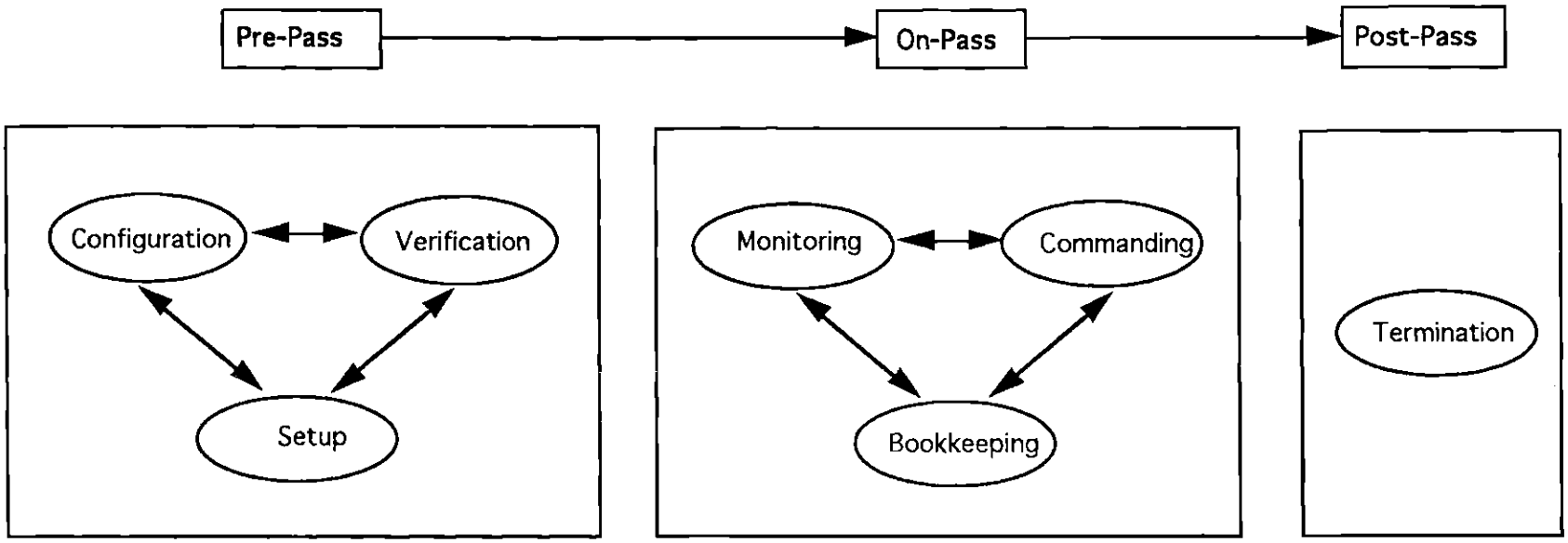


FIgure 6-22. The heterarchy of the GT-POCC operator function model

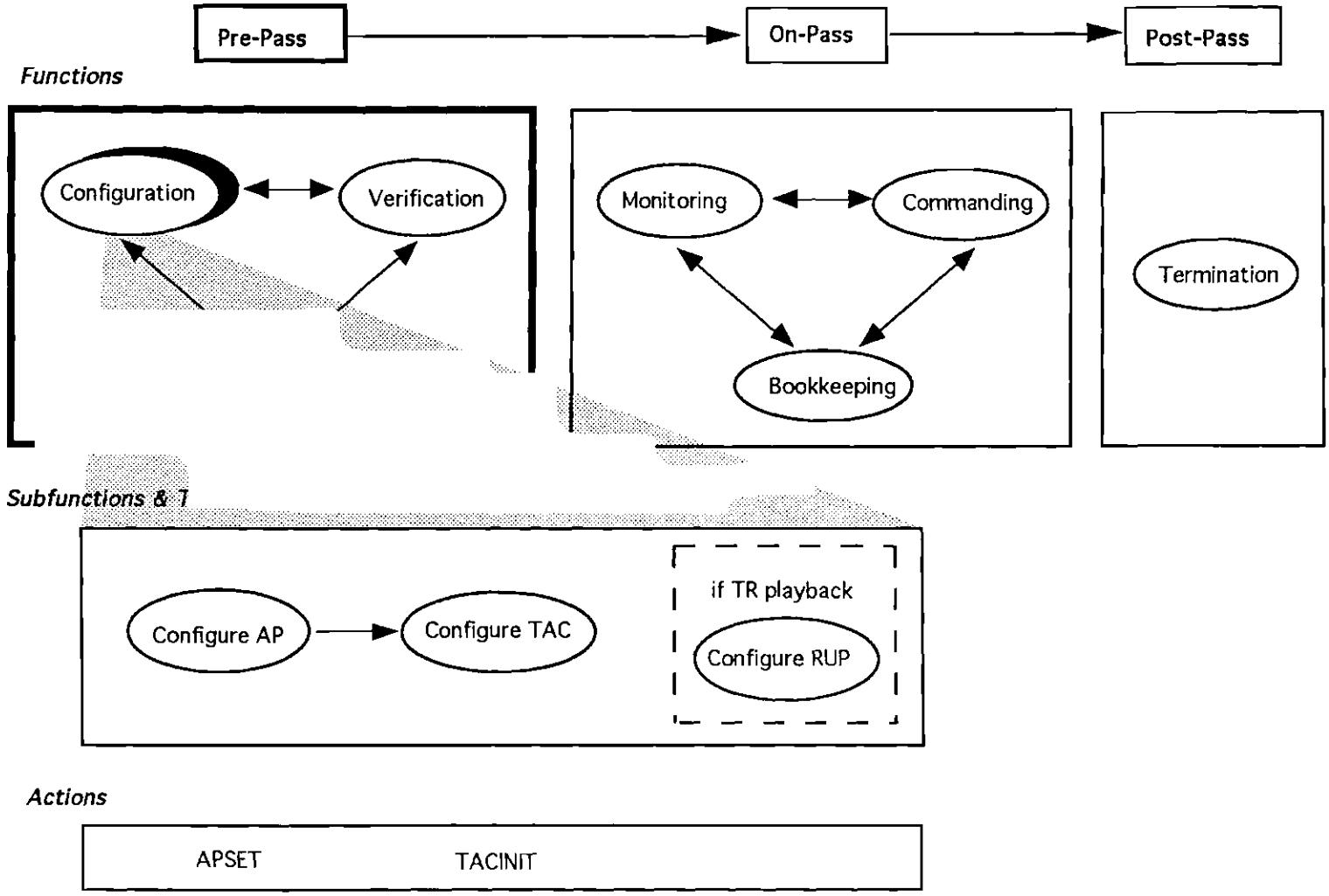


Figure 6-23. Decomposition of the GT-POCC Configuration function

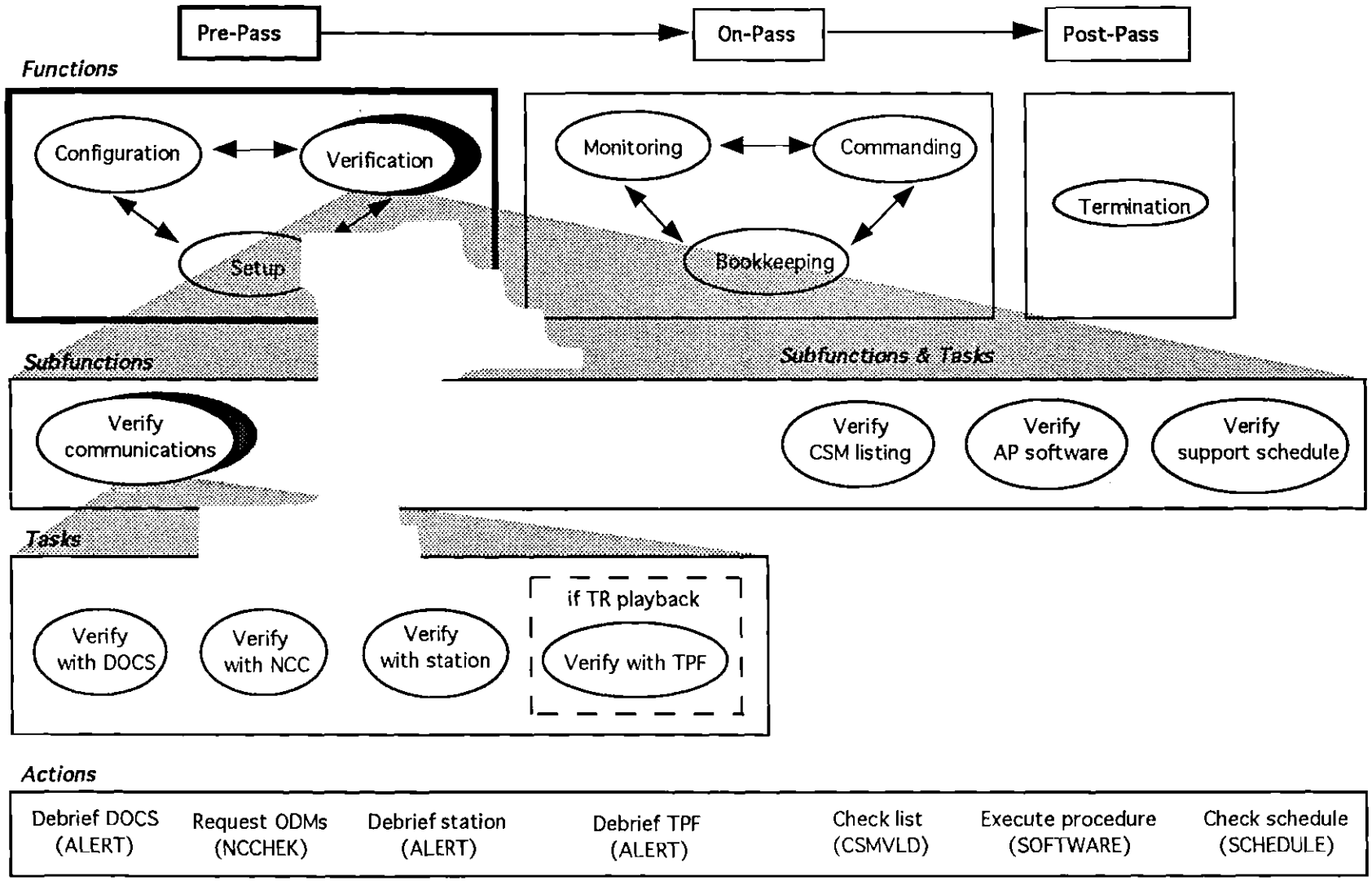


Figure 6-24. Decomposition of the GT-POCC Verification function

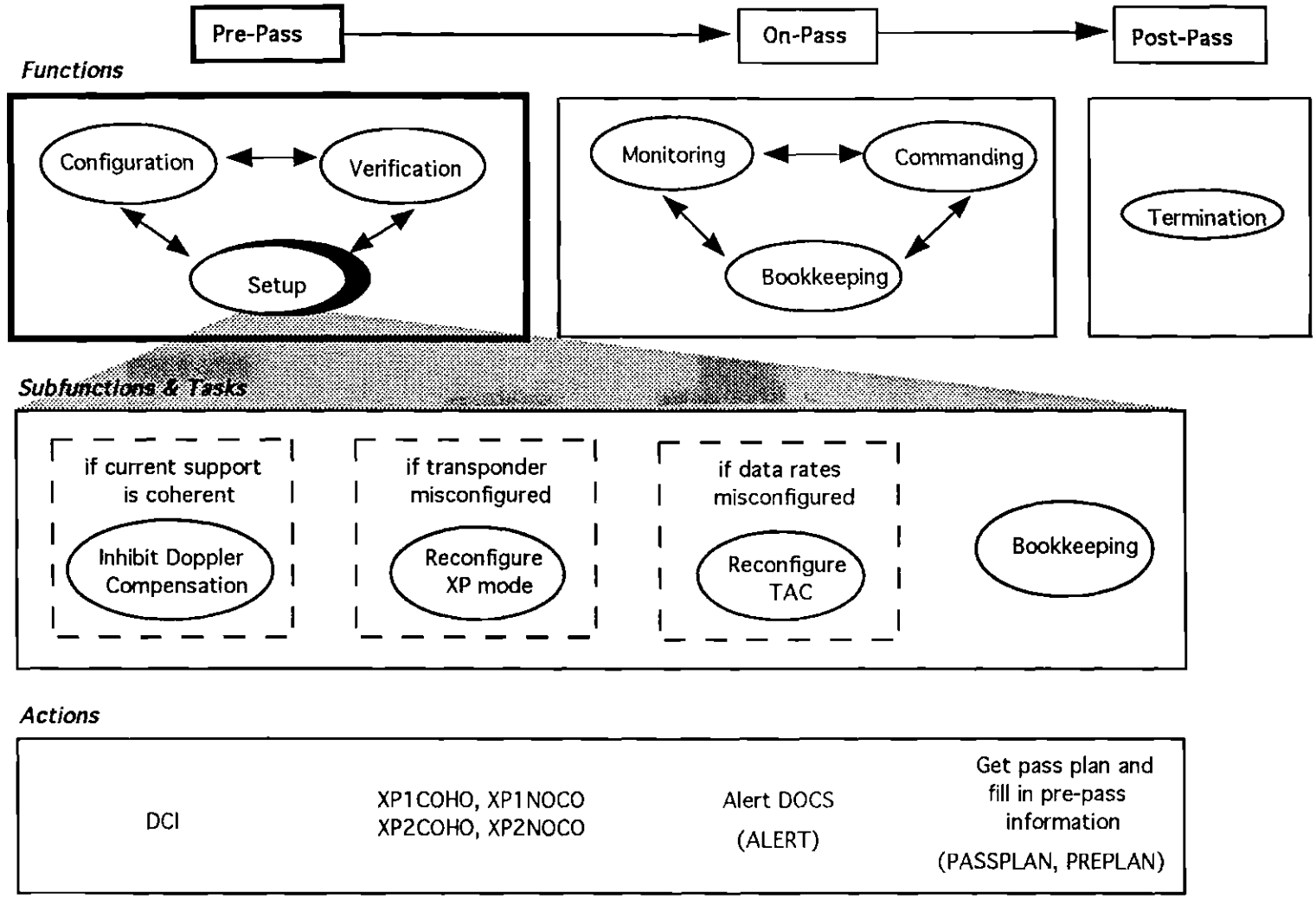


Figure 6-25. Decomposition of the GT-POCC Setup function

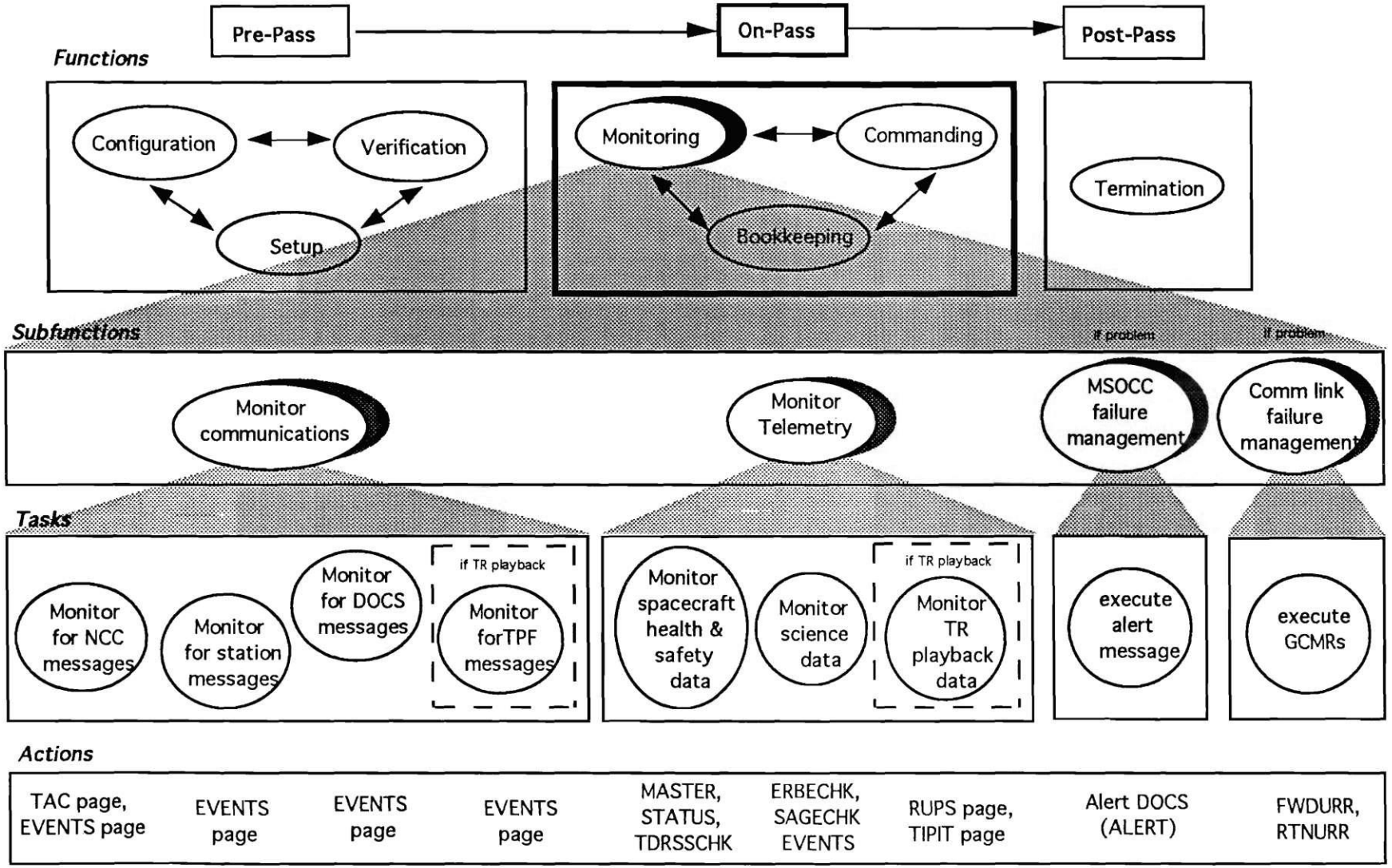


Figure 6-26. Decomposition of the GT-POCC Monitoring function

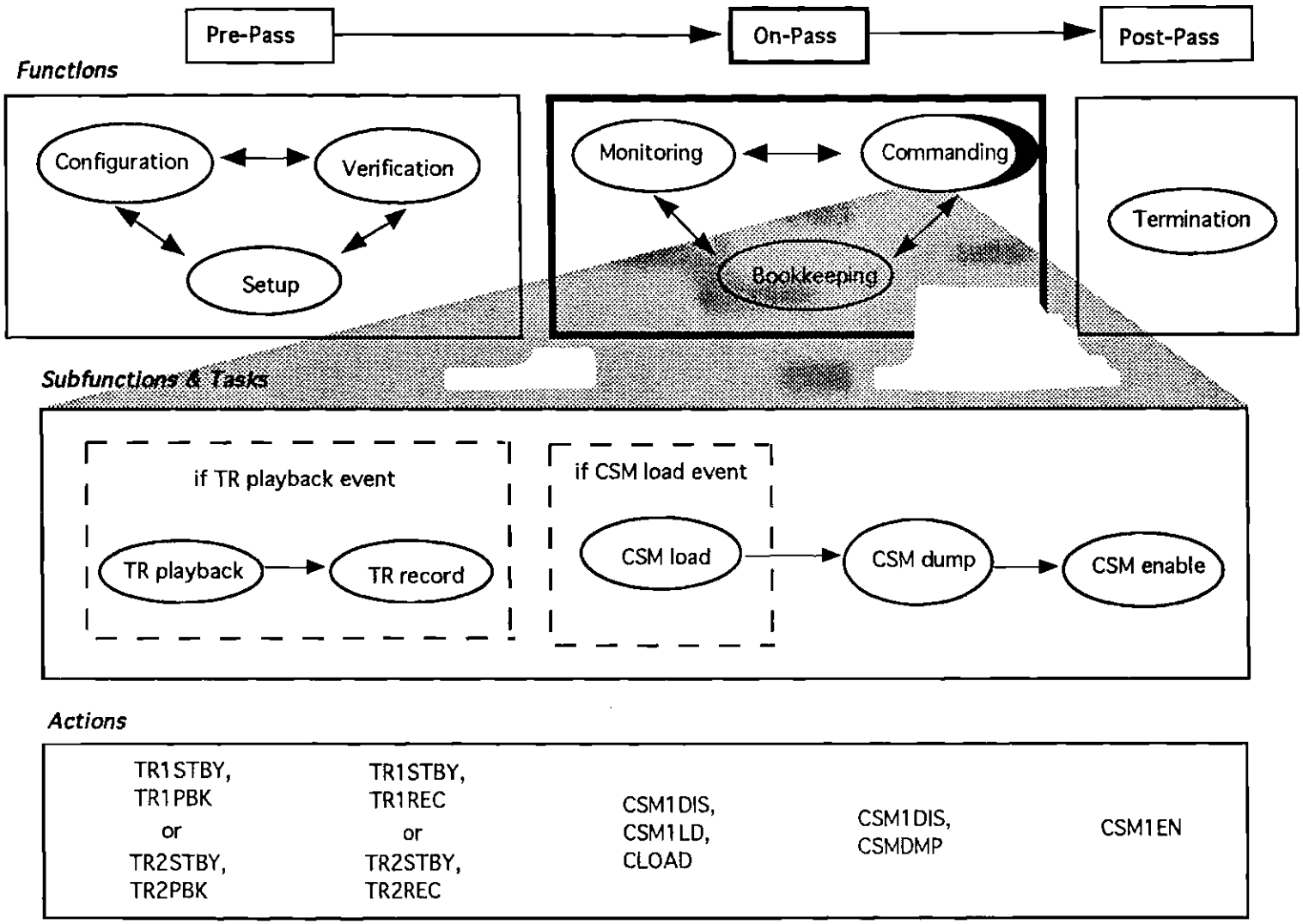


Figure 6-27. Decomposition of the GT-POCC Commanding function

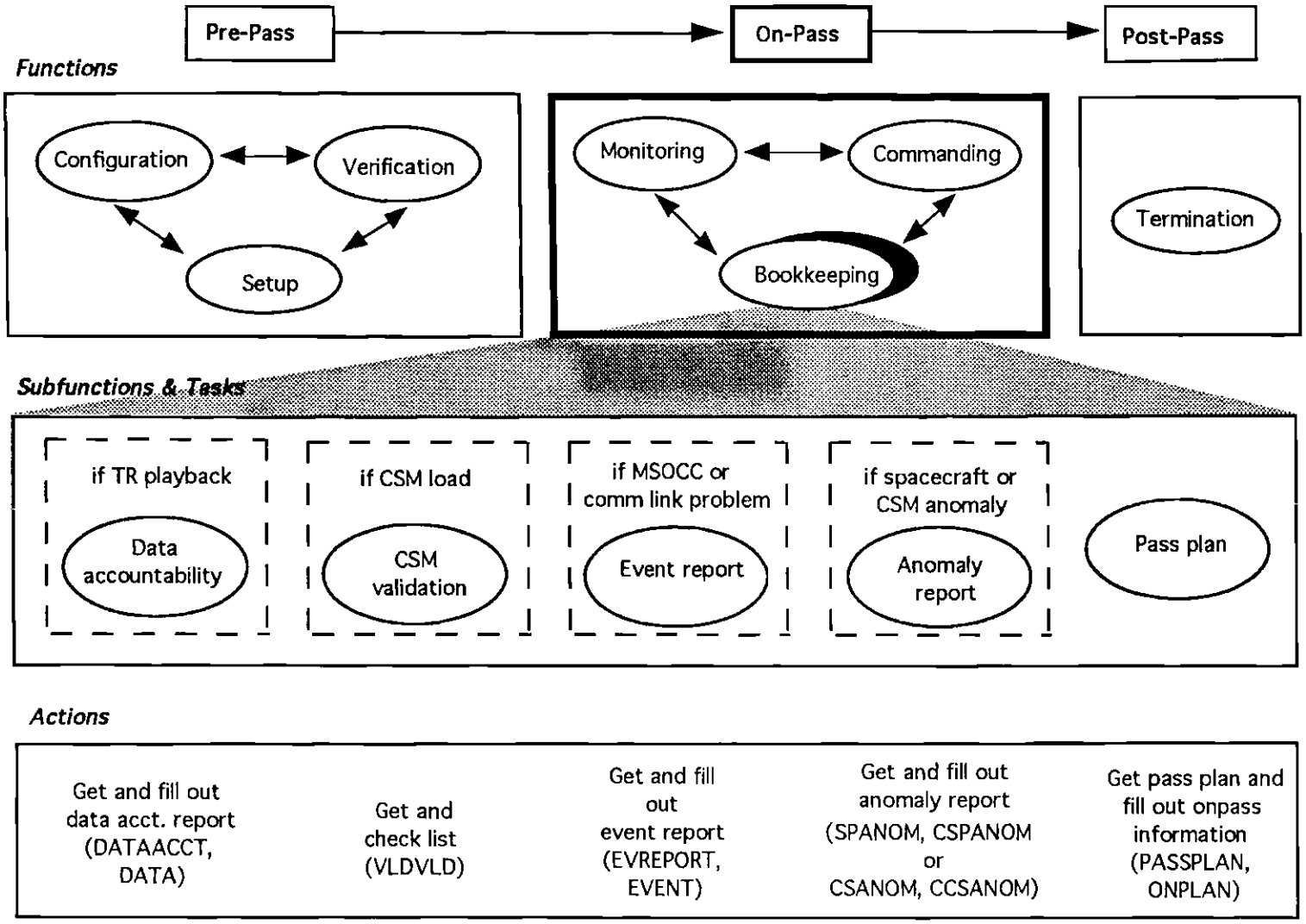


Figure 6-28. Decomposition of the GT-POCC Bookkeeping function

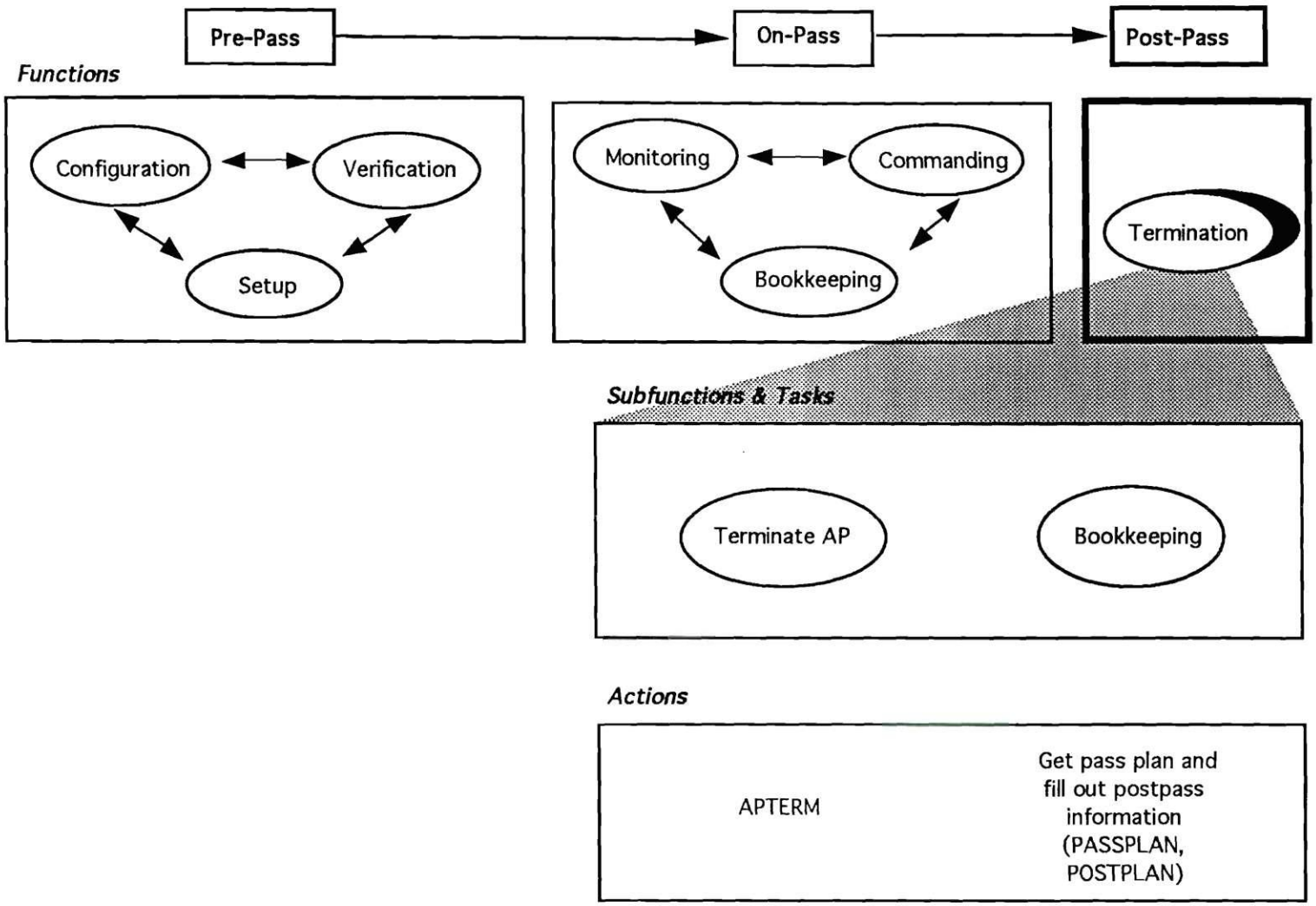


Figure 6-29. Decomposition of the GT-POCC Termination function

Table 6-2. The GT-POCC Operator Function Model

FUNCTIONS	SUBFUNCTIONS	TASKS	ACTIONS
CONFIG	C_AP (Config AP)	XC_AP (Execute)	APSET
	C_TAC (Config TAC)	XC_TAC (Execute)	TACINIT
VERIFY	VCOMM (Communications)	V_DOCS (With DOCS)	ALERT
		V_NCC (With NCC)	NCCHEK
		V_TNC (With TNC)	ALERT
		V_WS (With WS)	ALERT
		V_GN (With GN)	ALERT
		V_TPF (With TPF)	ALERT
	CSM (CSM listing)	XCSM (Execute)	CSMVLD
	SCHED (Support schedule)	XSCHED (Execute)	SCHEDULE
	SW (AP software)	XSW (Execute)	SOFTWARE
SETUP	DCI_MGT(DCI management)	XDCI (Execute DCI)	DCI
	COH_MGT (Coherency mgt.)	XCOH (Execute)	XPnCOHO/XPnNOCO
	RATE_MGT (Data rate mgt.)	XRATE (Execute)	RATE
	PASS (bookkeeping)	XPLAN (Pass plan)	PASSPLAN & PREPLAN

Table 6-2. The GT-POCC Operator Function Model (Cont'd)

MONITOR	MCOMM (Communications)	M_DOCS	EVENTS
		M_NCC	EVENTS, TACDISP
		M_TNC	EVENTS
		1_WS	EVENTS
		M_GN	EVENTS
		M_TPF	EVENTS
	TELEM (Telemetry data)	S/C (Spacecraft health and safety data)	MASTER, STATUS, TDRSSCHK, EVENTS
		SCIENCE (Scientific data)	ERBECHK, SAGECHK, EVENTS
		TR_DATA (Tape recorder data)	RUPSDISP, TIPIT
	FAIL_MGT (Failure mgt.)	XALERT (Execute)	ALERT
	R_GCMR	XRWDURR (Execute forwardlink GCMR)	RTNURR
	F_GCMR	XFWDURR (Execute forwardlink GCMR)	FWDURR
CMD Commanding	TRPBK (Tape recorder playback)	XTRPBK (Execute playback)	TRnSTBY & TRnPBK
	TRREC (Tape recorder record)	XTRREC (Execute record)	TRnREC

Table 6-2. The GT-POCC Operator Function Model (Cont'd)

CMD Commanding (Cont'd)	CSMLD (Command storage memory load)	XCSMLD (Execute)	CSM1DIS & CSM1LD & CLOAD
	CSMDMP (Command storage memory dump)	XCSMDMP(Execute)	CSM1DMP
	CSMEN (Command storage memory enable)	XCSMEN(Execute)	CSM1EN
BOOK Bookkeeping	PASS (pass plan)	XPLAN (Execute)	PASSPLAN & ONPLAN
	EVENT (Event report)	XEVENT (Execute)	EVREPORT & REPORT
	ANOM (Anomaly report)	XANOM (Execute)	CSPANOM, CCSMANOM
	B_TR (Bookkeeping for tape recorder playback)	DATA (Data accountability report)	DATAACCT & ACCOUNT
	B_CSMD (Bookkeeping for command storage memory dump)	XCSMVLD (Execute)	VLDVLD
TERM Termination	TERM_AP (Terminate AP)	XTERM_AP (Execute)	APTERM
	PASS (bookkeeping)	XPLAN (Pass plan)	PASSPLAN & POSTPLAN

CHAPTER VII

IMPLEMENTATION OF GT-VITA, THE GEORGIA TECH VISUAL INSPECTABLE TUTOR AND ASSISTANT FOR THE GEORGIA TECH PAYLOAD OPERATIONS CONTROL CENTER

PART I: SYSTEM AND TUTORIAL INTERFACES

Based on the operator function model and the enhanced OFMspert architecture discussed in Chapter V, as applied to the GT-POCC domain discussed in Chapter VI, GT-VITA (the Georgia Tech Visual Inspectable Tutor and Assistant) is developed for training novice operators of the GT-POCC environment. GT-VITA is a real-time, graphical and interactive intelligent tutoring system that interfaces with the simulated GT-POCC environment which consists of a GT-POCC simulation and a task interface. Theoretically speaking, after training, the simulated environment can be replaced by the actual task environment so that GT-VITA operates as an intelligent aiding system to the operator.

Figure 7-1 shows GT-VITA's interface configuration that spans across two display monitors. The interfaces consist of a domain system component and a tutorial dialog component. The domain system component is a graphical representation of the NASA data-information system that GT-VITA uses for instruction. The tutorial dialog component channels communication between GT-VITA and the student. In the following discussions, the two components are referred to as system interfaces and the tutorial interfaces respectively. Also, the word "system" is reserved for reference to the domain system. The GT-VITA system is often referred to as the "tutor".

The implementation of GT-VITA as an intelligent tutoring system is presented in three parts. First, this chapter describes GT-VITA from the perspective of "what it looks like". General features of the interface components are discussed. Next, Chapter VIII describes GT-VITA from the perspective of "how it

works" by presenting a detailed walkthrough of each lesson type to illustrate GT-VITA's pedagogy. Third, general implementation details from the design perspective are presented in Chapter IX followed by a discussion of GT-VITA with respect to the tutor/aid paradigm.

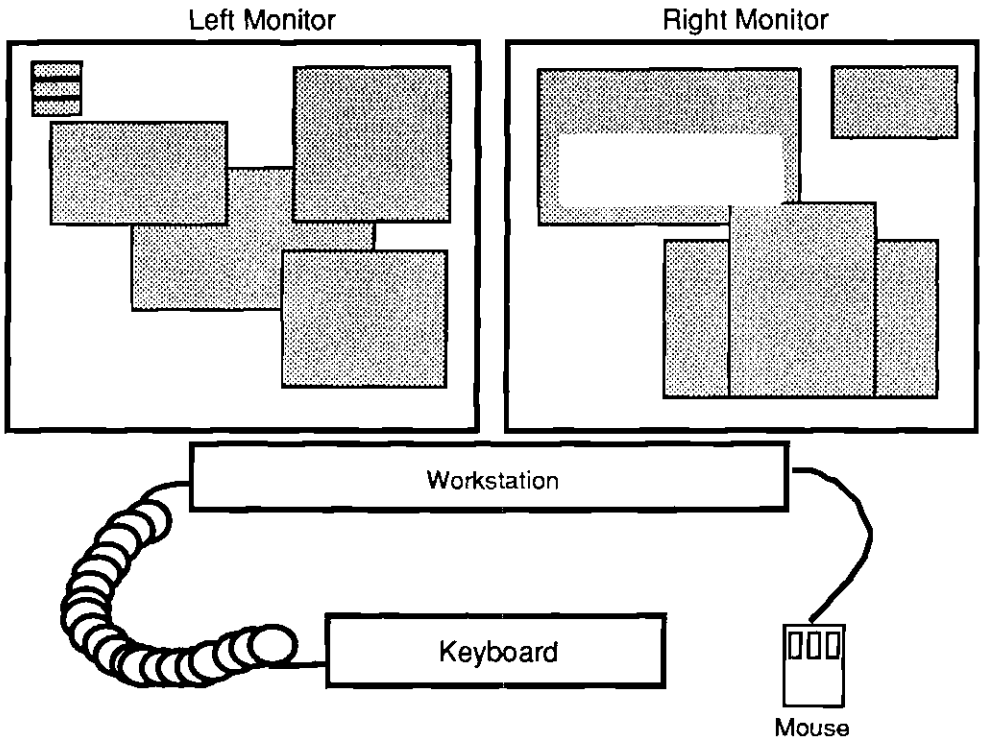


Figure7-1. The GT-VITA interface configuration.

GT-VITA System Interfaces

GT-VITA system interfaces serve three purposes: a means to visualize the domain system from various views and hierarchical levels, a means for operator information request, and a means for operator control of the system. Because most information request actions are embedded within the visualization elements, they are discussed under one heading.

Visualization of System

One problem identified in the analysis of knowledge requirements for the POCC was that operators do not have a broad, top-down perspective of the NASA system. Such a perspective has been argued to be crucial for appropriate learning of and reasoning about a system's normal behavior and its reactions to different types of failures (Goodstein et al., 1988; Williams et al., 1981). The GT-POCC graphical user interfaces were developed in direct response to these concerns.

GT-VITA provides a hierarchical, animated graphical view of the current state of the NASA data-information system. It allows a student to visualize system components, behavior and their relations at various levels of details. The student interacts with the graphical displays via direct manipulation of objects that represent system components. The operator clicks on objects to examine detailed views of the associated components. Data flows throughout the system, represented by arrows between objects, are animated during real-time supports at various levels of abstraction. Communication links between objects turn green to show an instance of data transmission between two objects, turn red if the data are degraded, and turn gray again when data transmission passes to the next object. Arrows that stop turning red or green indicate data lost between two objects.

Spacecraft components are displayed on the left monitor. All other NASA components supporting the spacecraft are displayed on the right monitor. Each graphical display (called a panel) is described next; those on the right monitor are discussed first followed by those on the left.

NASA Data/Information System. This panel shows a high-level view of the entire NASA data/information system. Major components of the NASA system including a spacecraft, a TDRS system, a Ground Network, White Sands/NASA ground terminal and Goddard Space Flight Center are represented as objects (Figure 7-2). This panel is the main panel from which all other graphical displays are accessible. Except for the White Sands object, all other objects in this panel are clickable. For visualization, a clickable object provides a decomposition of the component the object represents. In other words, the object is inspectable. For operator control, a clickable object displays the control panel associated with the component the object represents. Inspectable objects are discussed in this section. The second type of objects are discussed in a subsequent section.

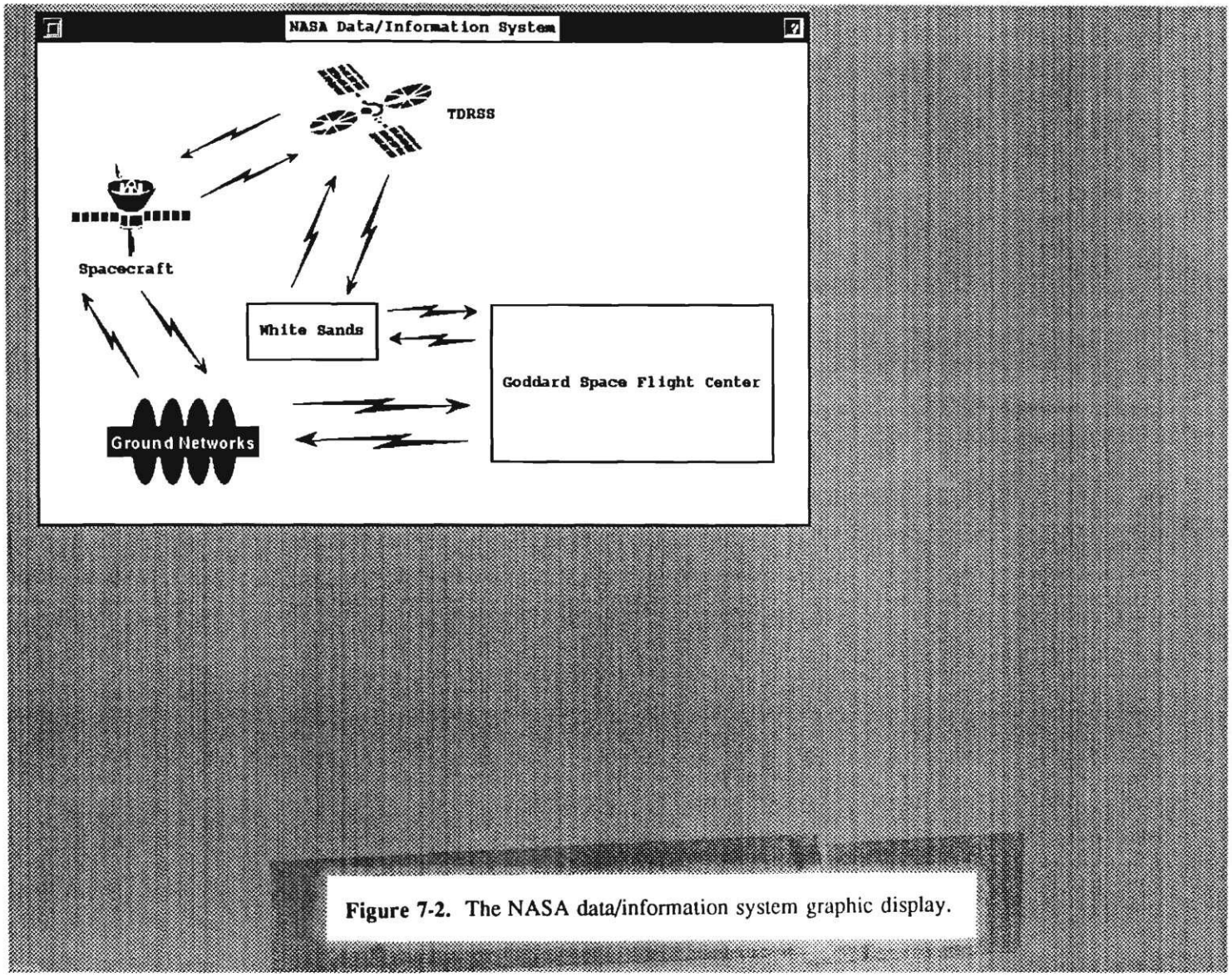


Figure 7-2. The NASA data/information system graphic display.

Goddard Space Flight Center. When the student clicks on the Goddard Space Flight Center object, a schematic view of GSFC is shown. The component objects of GSFC are the collection of NASA communication lines (NASCOM), Network Control Center (NCC), Science Data Processing Facility (SDPF), and MultiSatellite Operations Control Center (MSOCC) (Figure 7-3). All objects are clickable except for NASCOM.

Telemetry Processing Facility. The student clicks on the SDPF object on the NASA Data/Information System panel to inspect the amount of data accumulating at the Telemetry Processing Facility during a tape recorder playback activity (Figure 7-4). The action to inspect the SDPF is equivalent to the action to request the TIPIT display page on task interface (Figure 6-12).

TDRSS Support. When the student clicks on the TDRSS object on the NASA Data/Information System panel, a view of the NASA components that comprise a TDRSS support is shown (Figure 7-5). the spacecraft and the MSOCC objects are inspectable.

Ground Network Support. When the student clicks on the TDRSS object on the NASA Data/Information System panel, a view of the NASA components that comprise a TDRSS support is shown (Figure 7-6). The spacecraft object and the MSOCC object are inspectable.

MultiSatellite Operations Control Center. When the student clicks on the MSOCC object in one of these panels: Goddard Space Flight Center, TDRSS Support or Ground Network Support, a detailed view of the MultiSatellite Operations Control Center is shown. The MSOCC facility consists of a Telemetry and Command Computer (TAC), an Application Processor (AP) and a Recorder Utility Processor System (RUP) (Figure 7-7). The NASA Communications object is also shown to show data input and output to the MSOCC components.

During a real-time support, besides the data flow representation between objects, each component can be further characterized by its state and the quality of data flowing through it. The border color of the box represents component state; the small circles inside each box represent data flow. A white border indicates an idle component in working condition. A green border indicates a busy component in working condition configured for the current support. A red border indicates a component with a hardware failure. The blue circles inside the box turn green at each instance of good data transmission. The circles turn red if data are

degraded by software or transmission problems. In case of a hardware problem, data flow to subsequent components is terminated.

Only the TAC object can be inspected further. However, the label buttons (a box around a label above an object) for TAC, AP and RUP are clickable for control purposes.

Telemetry And Command Computer. When the TAC object on the MSOCC panel is clicked upon, a detailed view of the individual channels that comprise the Telemetry and Command computer is displayed. This panel shows the five dedicated channels (three "A" channels and two "B" channels) inside the TAC computer. The view displays the various data flows entering and leaving the MSOCC facility that are processed by the channels (Figure 7-8). A failure at the TAC level usually implies one or more failures at the channel level. The channels are characterized in the same way as the MSOCC display. The action to inspect the TAC object is equivalent to the action to request the TAC alphanumeric display page on the command panel (Figure 6-10). Instead of monitoring the data flow quantitatively, the student learns about the behavior of TAC qualitatively with graphical objects and data flows.

Recorder Utility Processor System. The student clicks on the button labelled "Recorder Utility Processor System" above the RUP object on the MSOCC graphic panel to inspect the amount of data being recorded on backup tapes during a spacecraft tape recorder playback activity (Figure 7-9). The action to click on the RUP label button is equivalent to the action to request the RUPS display page on the task interface (Figure 6-11).

Spacecraft Subsystems. When the student clicks on the spacecraft object in one of these panels: the NASA Data/Information System, the TDRSS Support or the Ground Network Support, a detailed view of the Spacecraft Subsystems is shown (Figure 7-10). On this panel, only two arrows between the Command & Data Handling Subsystem object and the Radio Frequency Communications Subsystem object are shown in grey to indicate data flow exclusively during a real-time support. The other arrows are shown in white to indicate the continuous power supply from the Power Subsystem. Each subsystem object can be clicked upon to view the spacecraft components that comprise the subsystem.

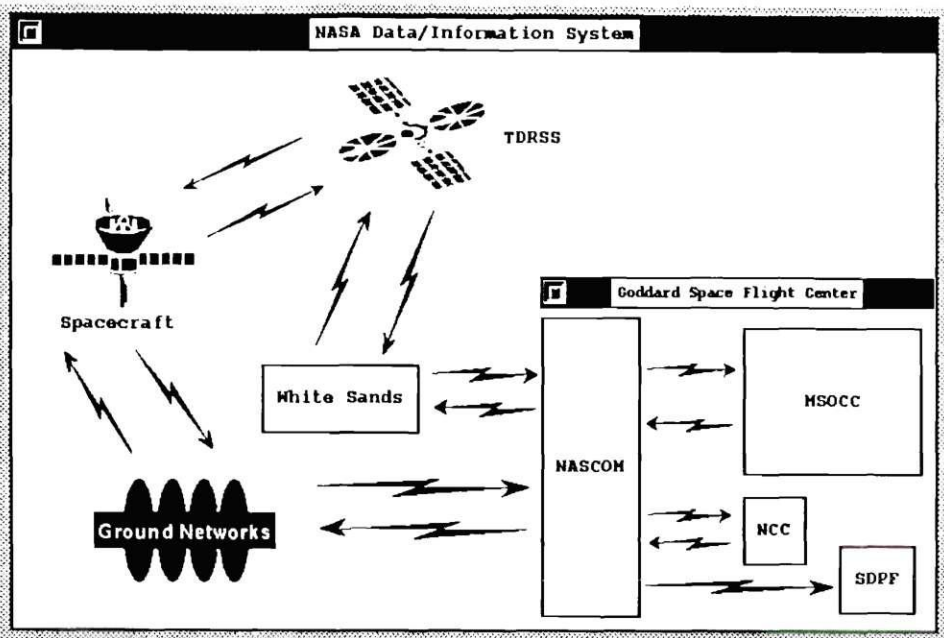


Figure 7-3. The Goddard Space Flight Center graphic display.

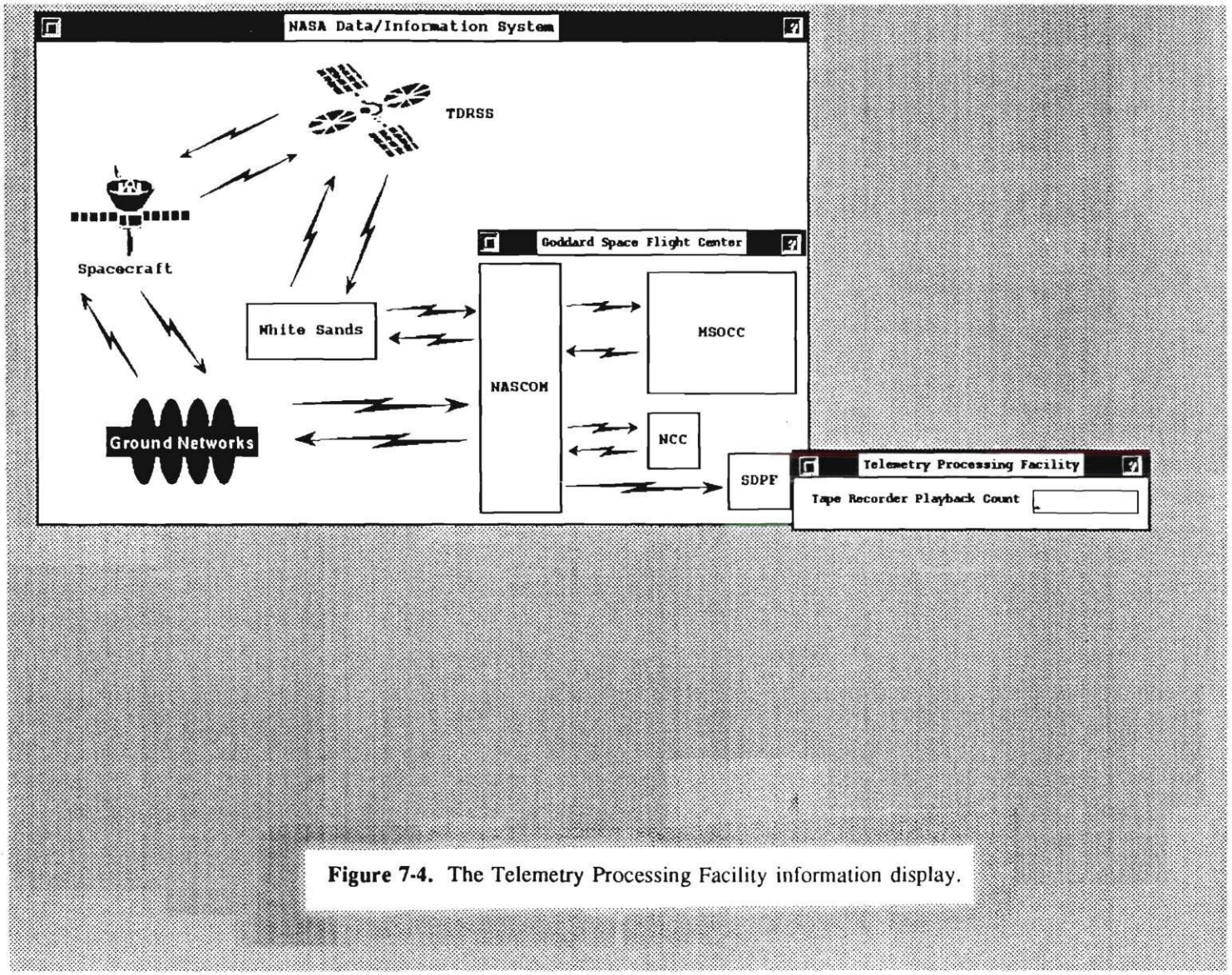


Figure 7-4. The Telemetry Processing Facility information display.

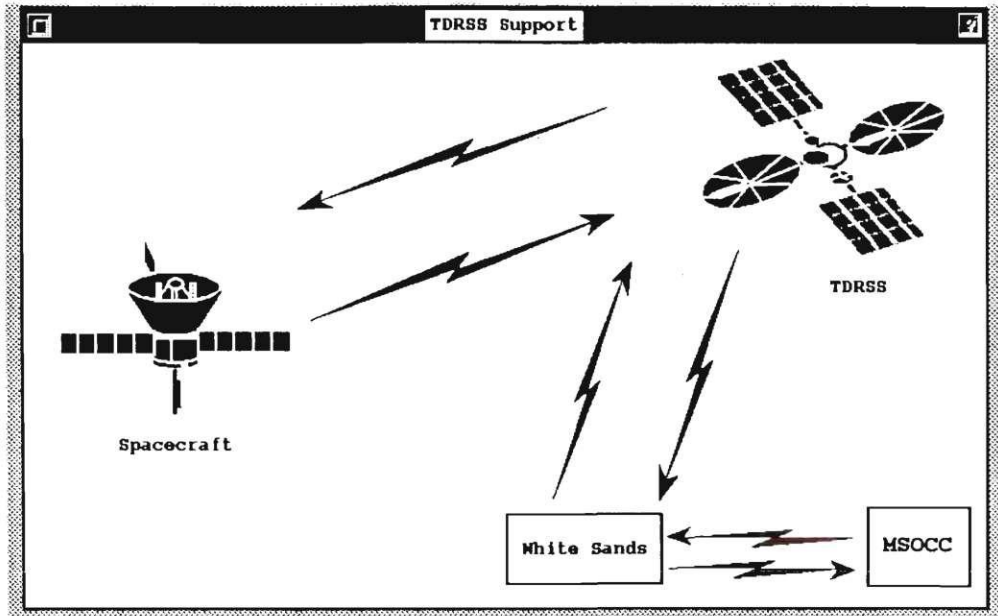


Figure 7-5. The TDRSS support graphic display.

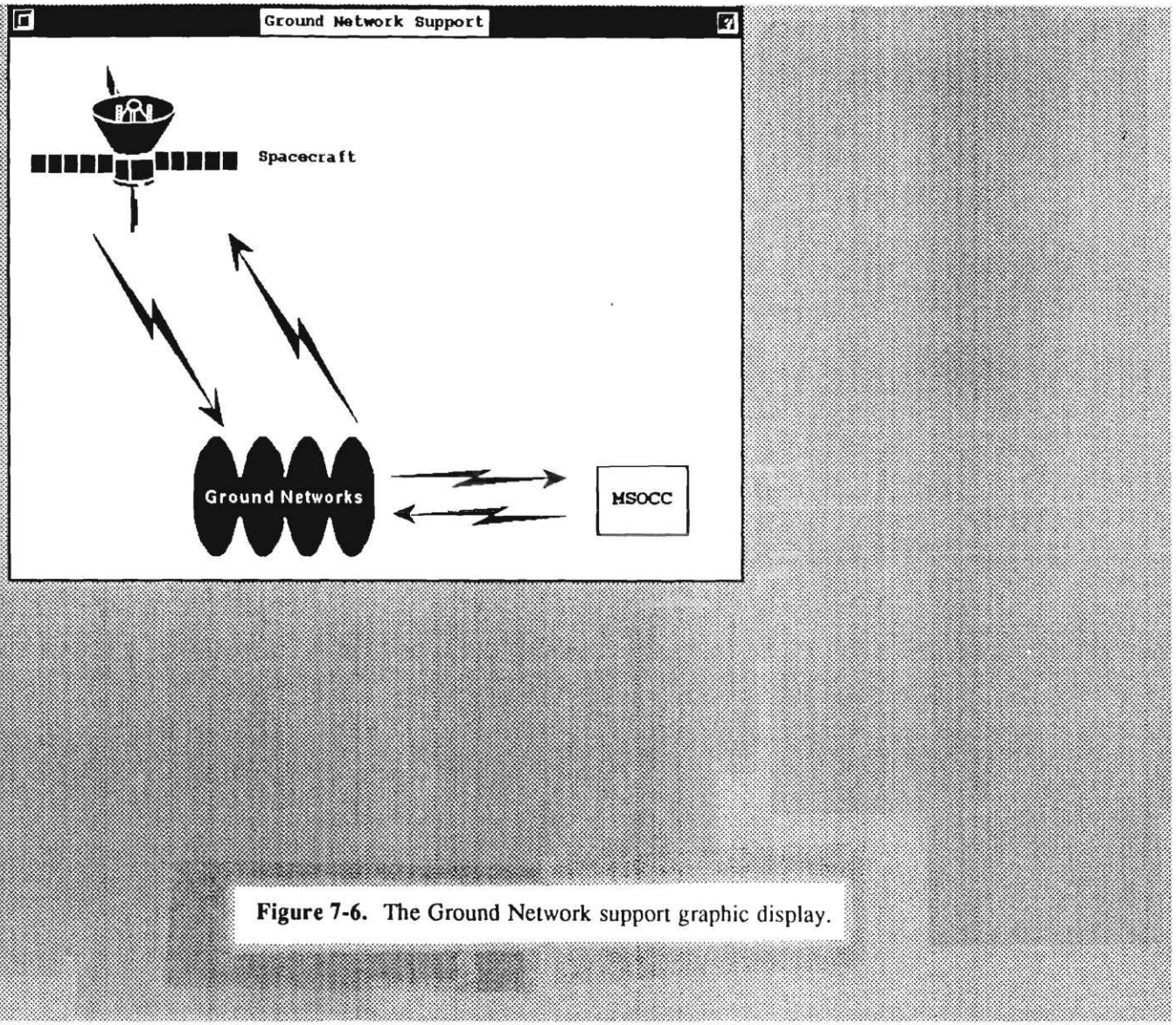


Figure 7-6. The Ground Network support graphic display.

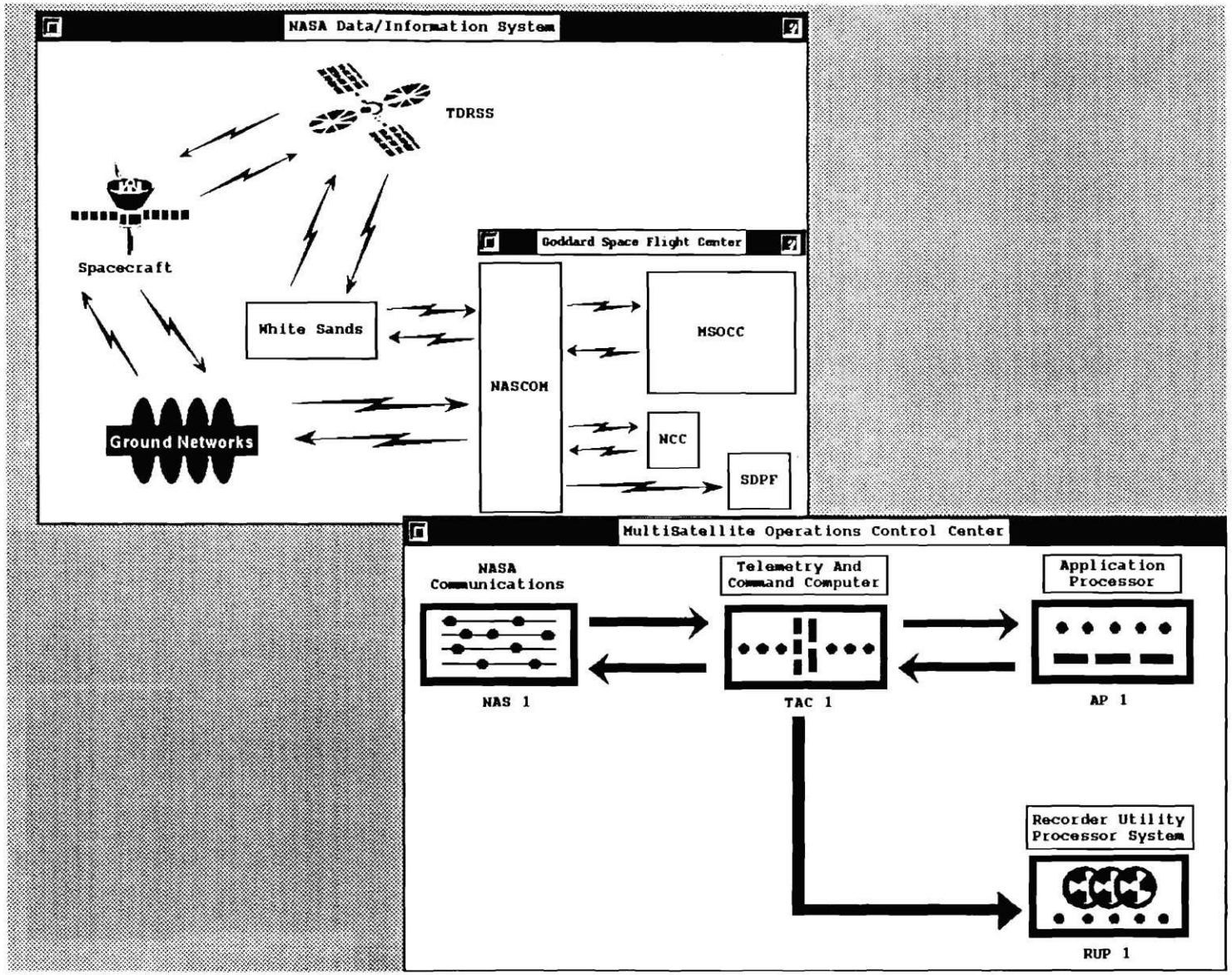


Figure 7-7. The MultiSatellite Operations Control Center (MSOCC) graphic display.

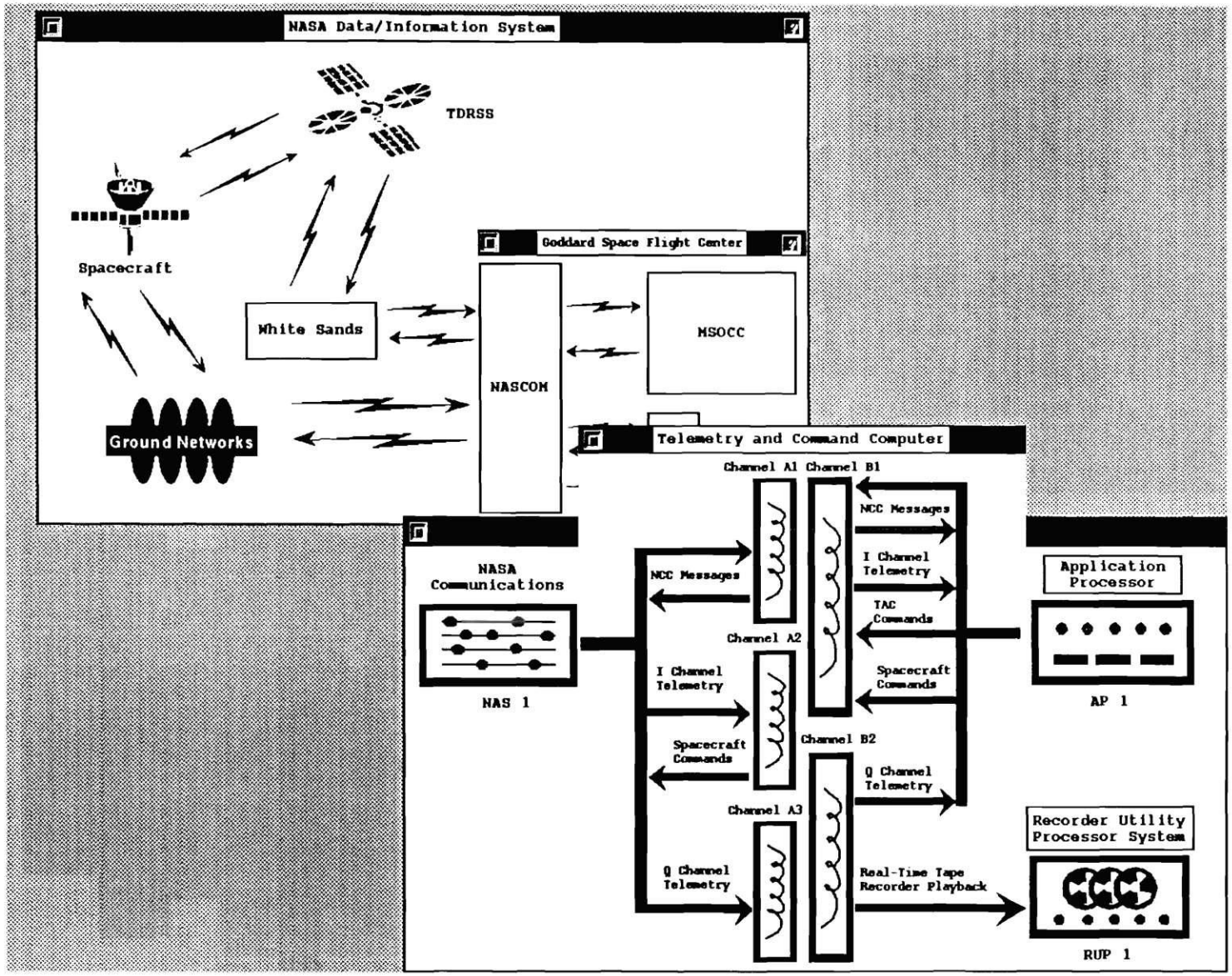


Figure 7-8. The Telemetry and Command Computer (TAC) graphic display.

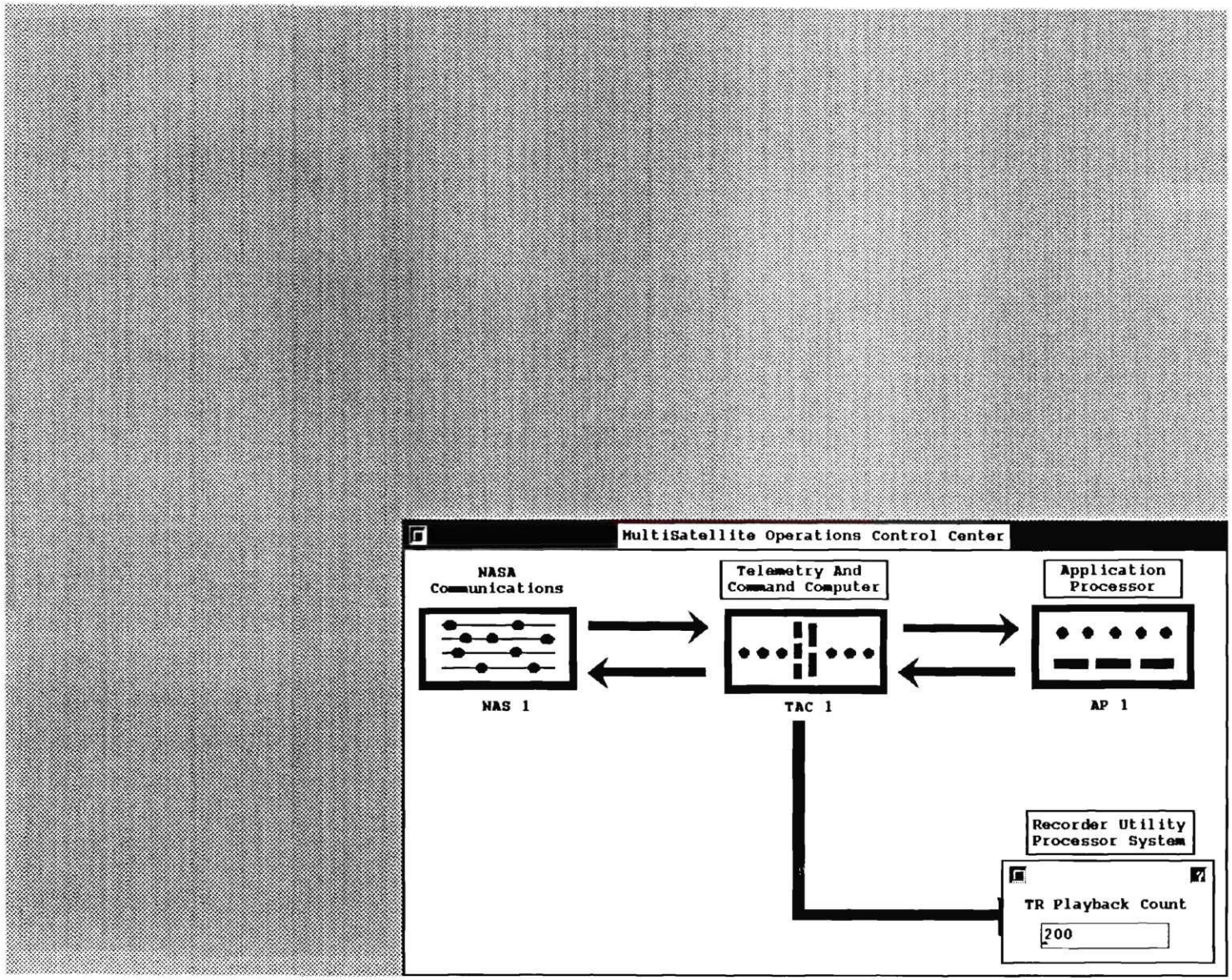


Figure 7-9. The Recorder Utility Processor System (RUP) information display.

Attitude Control Subsystem. This panel shows a simplified view of the Attitude Control Subsystem with three static objects representing gyroscopes in the x, y and z directions (Figure 7-11). Each gyroscope has an angle, a real parameter which is updated dynamically and continuously (i.e., regardless of a real-time support). None of the objects on this panel can be inspected further.

Power Subsystem. This panel shows static pictures of two Solar Arrays, two batteries, the essential bus and the non-essential bus (Figure 7-12). The solar arrays have no parameters. The batteries have real parameter values for charge and charge/discharge ratios. The buses have real parameter values for voltage and current. All these parameters are updated dynamically and continuously. None of the objects here are inspectable.

Command and Data Handling Subsystem. This panel shows two scientific instruments (ERBE and SAGE), two Telemetry and Command Units, two tape recorders and the command storage memory (Figure 7-13). The data links between a science instrument, a Telemetry and Command Unit and a tape recorder form the scientific data collection path that is animated continuously. For each unit of data collected and transmitted, the links turn green successively from a scientific instrument, to the corresponding Telemetry and Command Unit, and to the corresponding tape recorder. Furthermore, the stripchart inside the science instrument turns from blue to green to indicate data collection. Some circles inside the Telemetry and Command Unit turns from blue to green to indicate data processing. Finally, the tape recorder "turns" to indicate data recording.

During a real-time support, all spacecraft data downlinked to the ground (e.g., telemetry or tape recorder playback data) and all ground data uplinked to the spacecraft (e.g., spacecraft commands) are processed by a Telemetry and Command Unit. Thus, data flow between the unit and the Radio Frequency Subsystem is also animated. On this panel, the tape recorder and the command storage memory objects are inspectable.

Command Storage Memory. This panel displays the internal structure of the command storage memory that is partitioned into a normal memory and a block memory (Figure 7-14). The normal memory shows a set of spacecraft commands scheduled to be executed in absolute times. The block memory shows a set of "canned" commands in relative time. As each command is executed in the normal memory, the

command is highlighted. Often, the normal memory has a "jump" command that invokes a specified set of commands in the block memory. In that case, the commands in the block memory are highlighted as they are executed, after which the highlighting resumes at the normal memory. Thus, the student can visualize the dynamics between the normal and block memory.

Instead of viewing the command execution of memory partitions, the student has the option to query each command for further explanation. However, the explanations are not available at the moment. A viable future research project is to explore an inspectable and runnable command storage memory that not only provides explanations, but also simulates and visualizes the effects of a command on the spacecraft.

This panel also shows the current status of the command storage memory in terms of its mode and pending memory load to be uplinked from the ground. The control features of this panel are discussed in a later section.

Radio Frequency Subsystem. When the student clicks on the Radio Frequency Subsystem object on the Spacecraft Subsystems panel, a detailed view shows the components for this subsystem: two transponders and two antennas (Figure 7-15). The border of a transponder turns green when it is turned on for a real-time support. The coherency of the transponder is also shown inside the box. For each instance of data transmission to and from the ground, the arrows connecting the transponder and its corresponding Telemetry and Command Unit in the Command & Data Handling Subsystem turn green. The transponder is further characterized by its signal strength (AGV). The antenna also turns green when transmitting or receiving data. All data enter or leave the spacecraft through the antenna. The antenna is characterized by its angle. Both the AGV and angle values are real parameters that are activated and updated dynamically during real-time supports only. The transponder and antenna objects are clickable.

The displays for the Attitude Control Subsystem, the Power Subsystem, the Command & Data Handling Subsystem, and the Radio Frequency Subsystems are equivalent to the MASTER display page and STATUS procedure on the command panel (Figure 6-6); parameters for each subsystem are dynamically updated in the same ways as in the MASTER panel and the STATUS procedure.

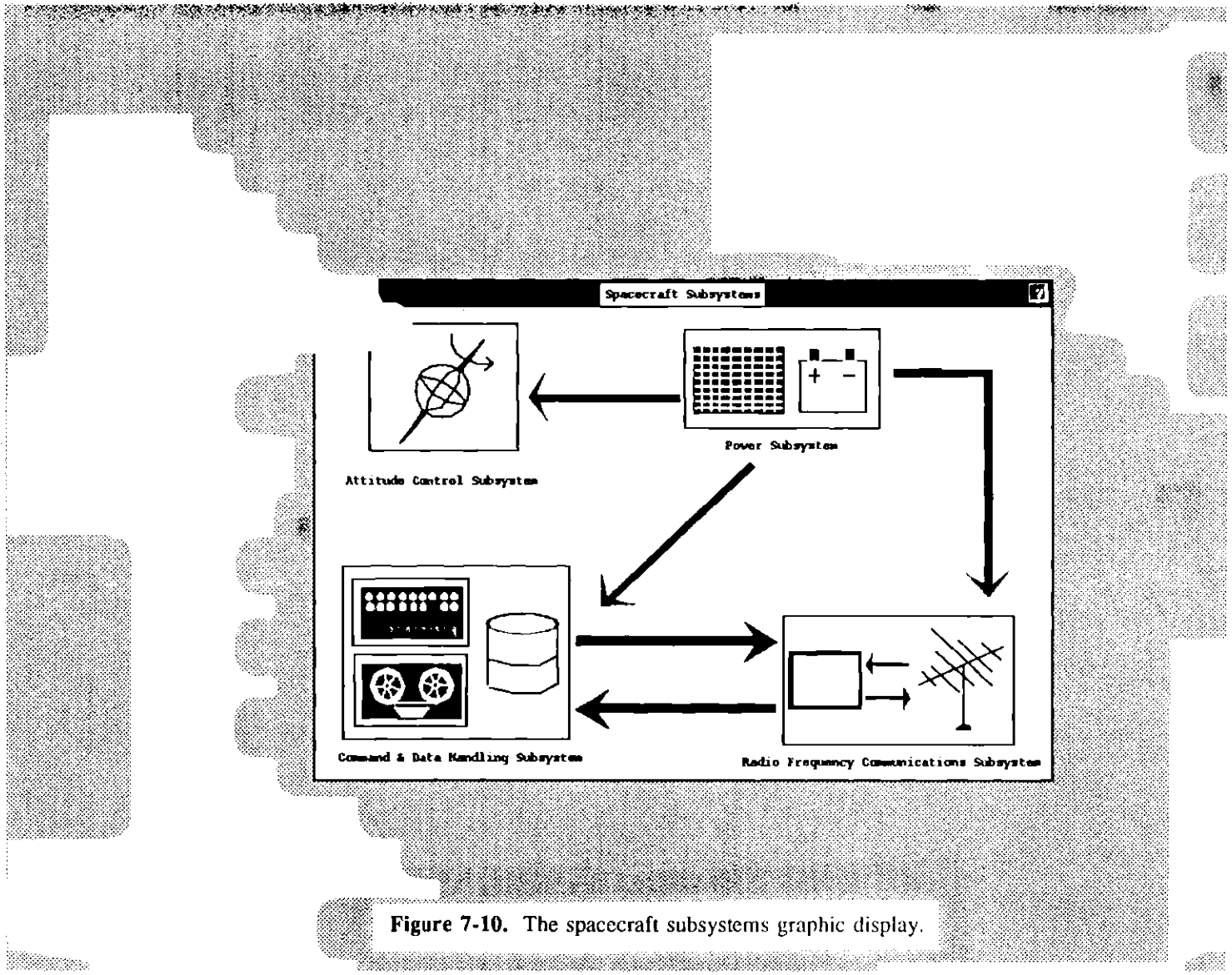


Figure 7-10. The spacecraft subsystems graphic display.

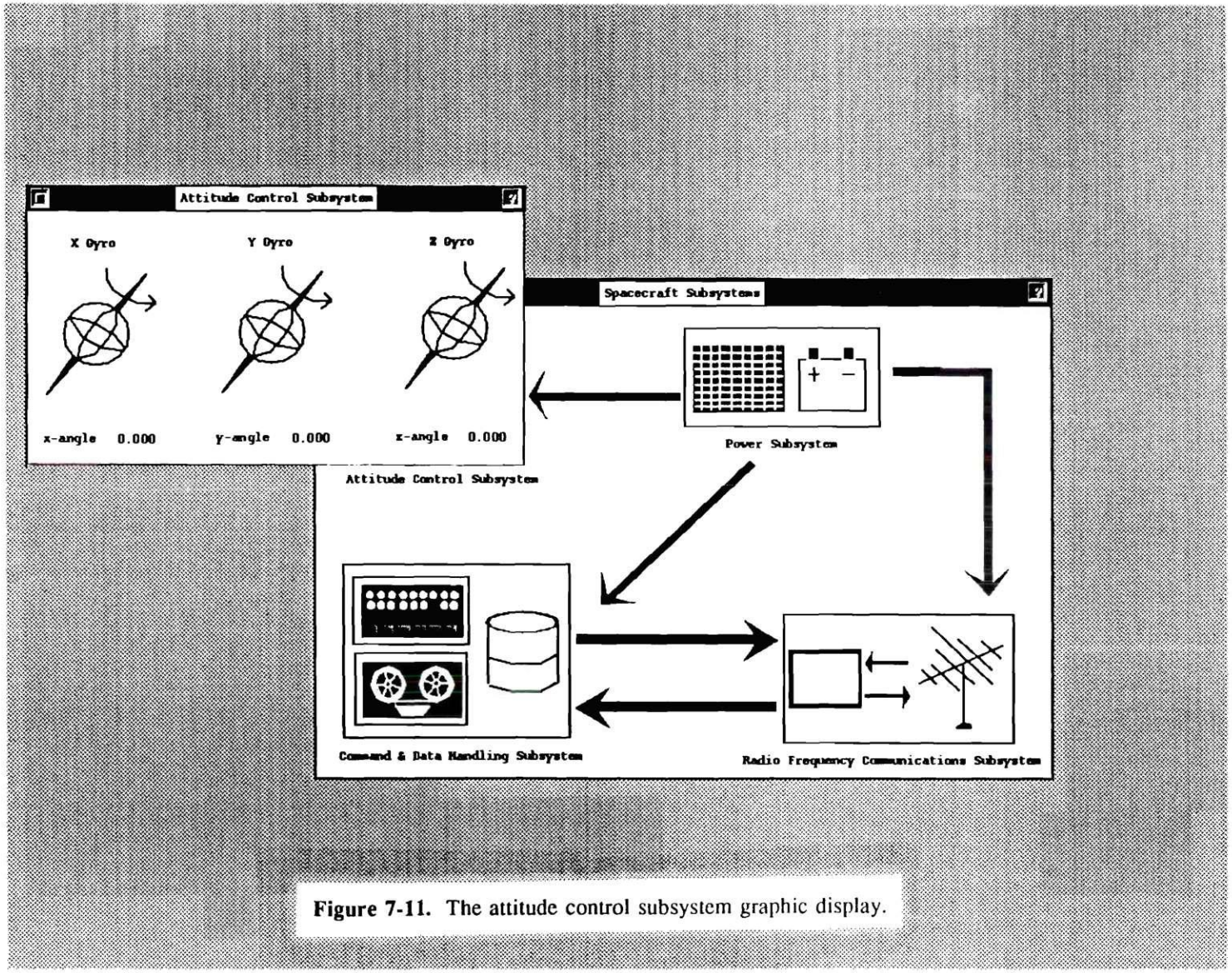


Figure 7-11. The attitude control subsystem graphic display.

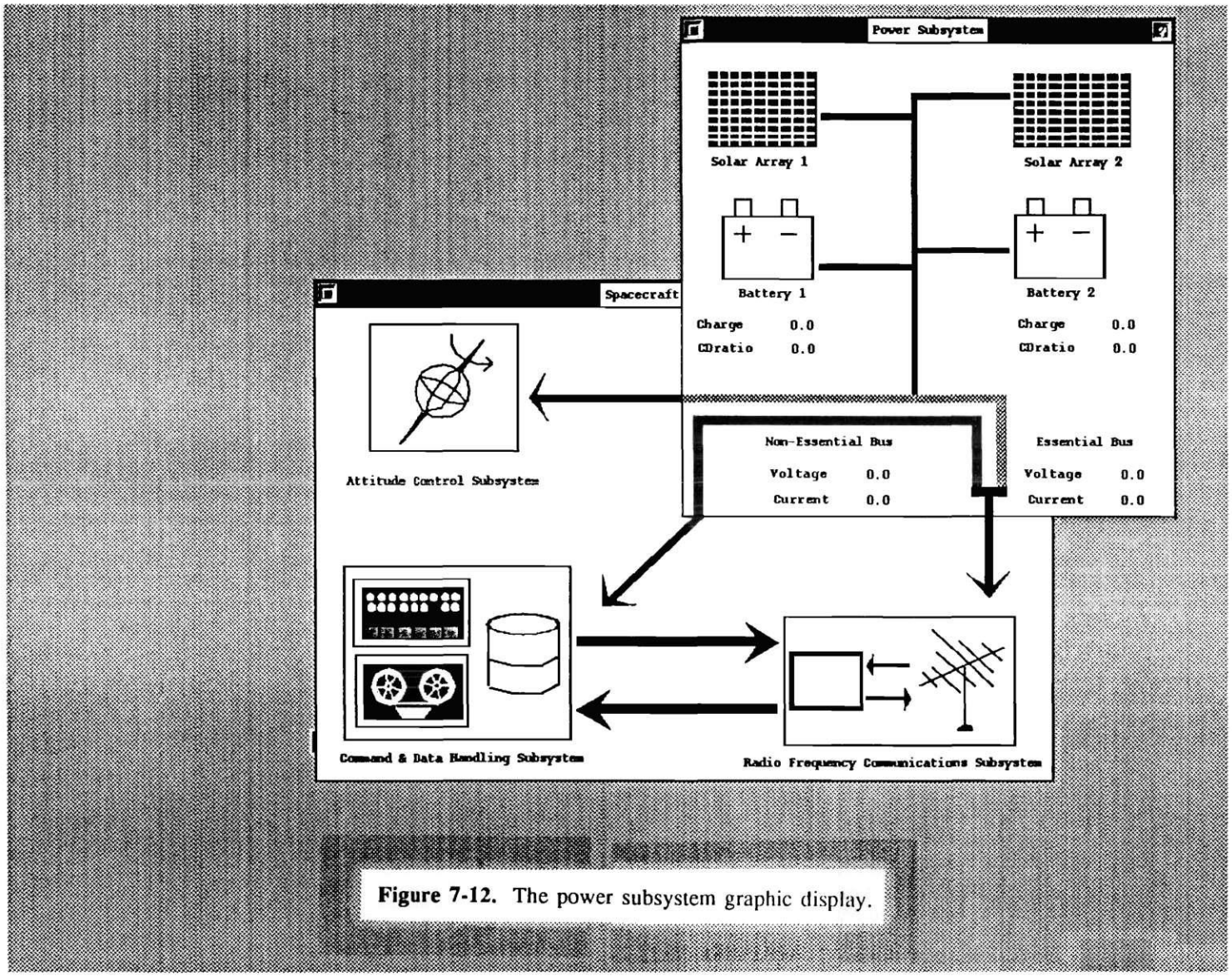


Figure 7-12. The power subsystem graphic display.

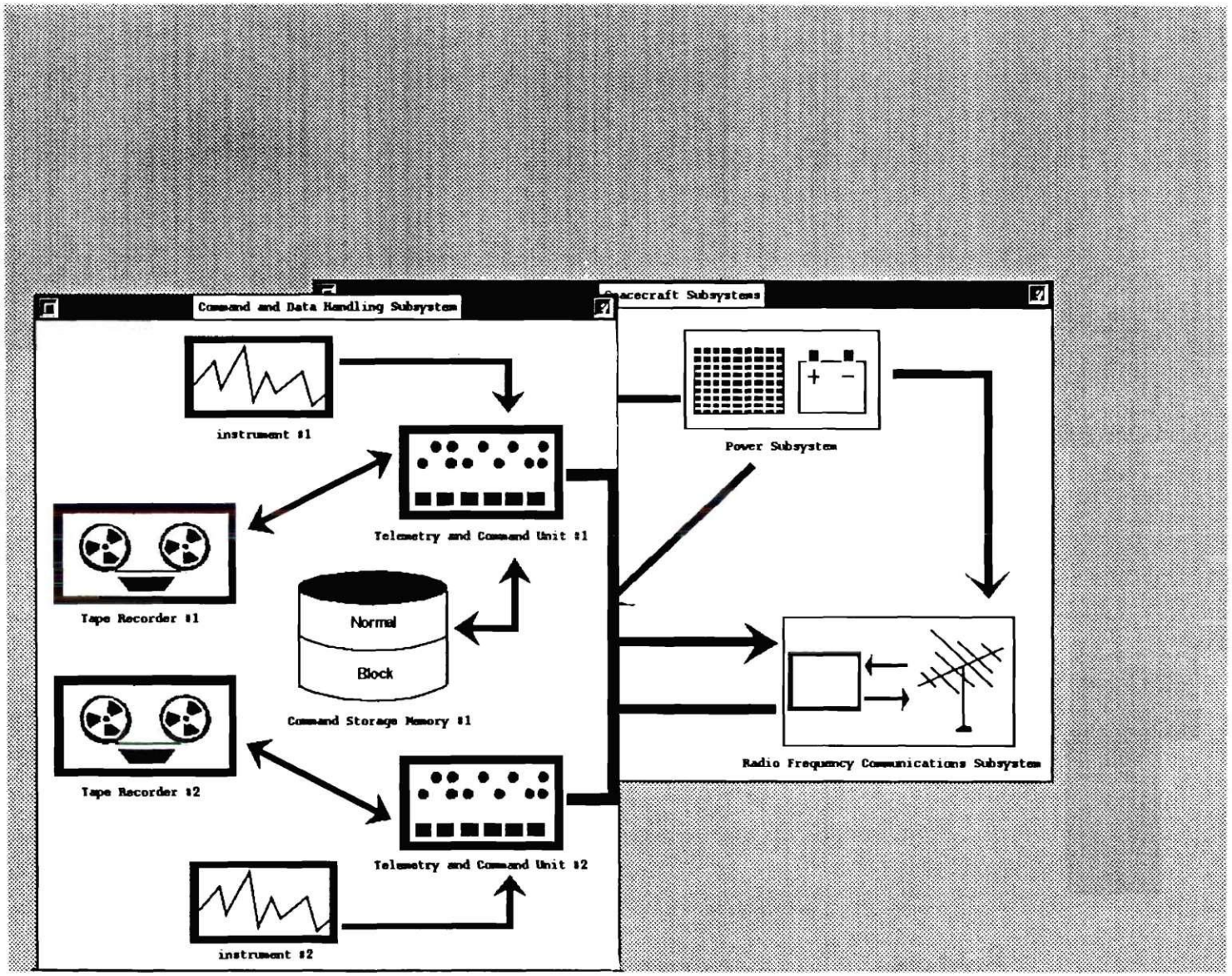


Figure 7-13. The command and data handling Subsystem graphic display.

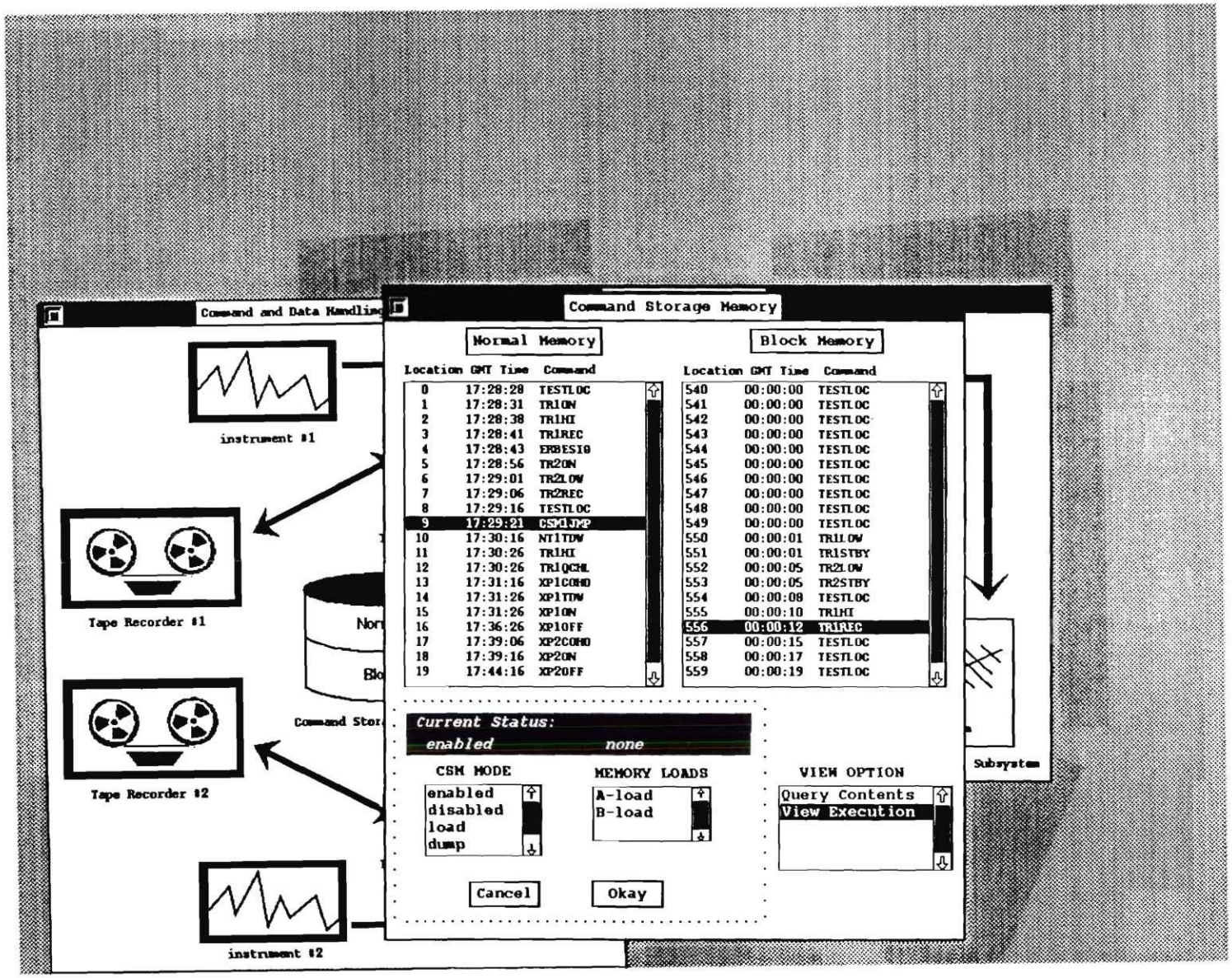


Figure 7-14. The command storage memory graphic display.

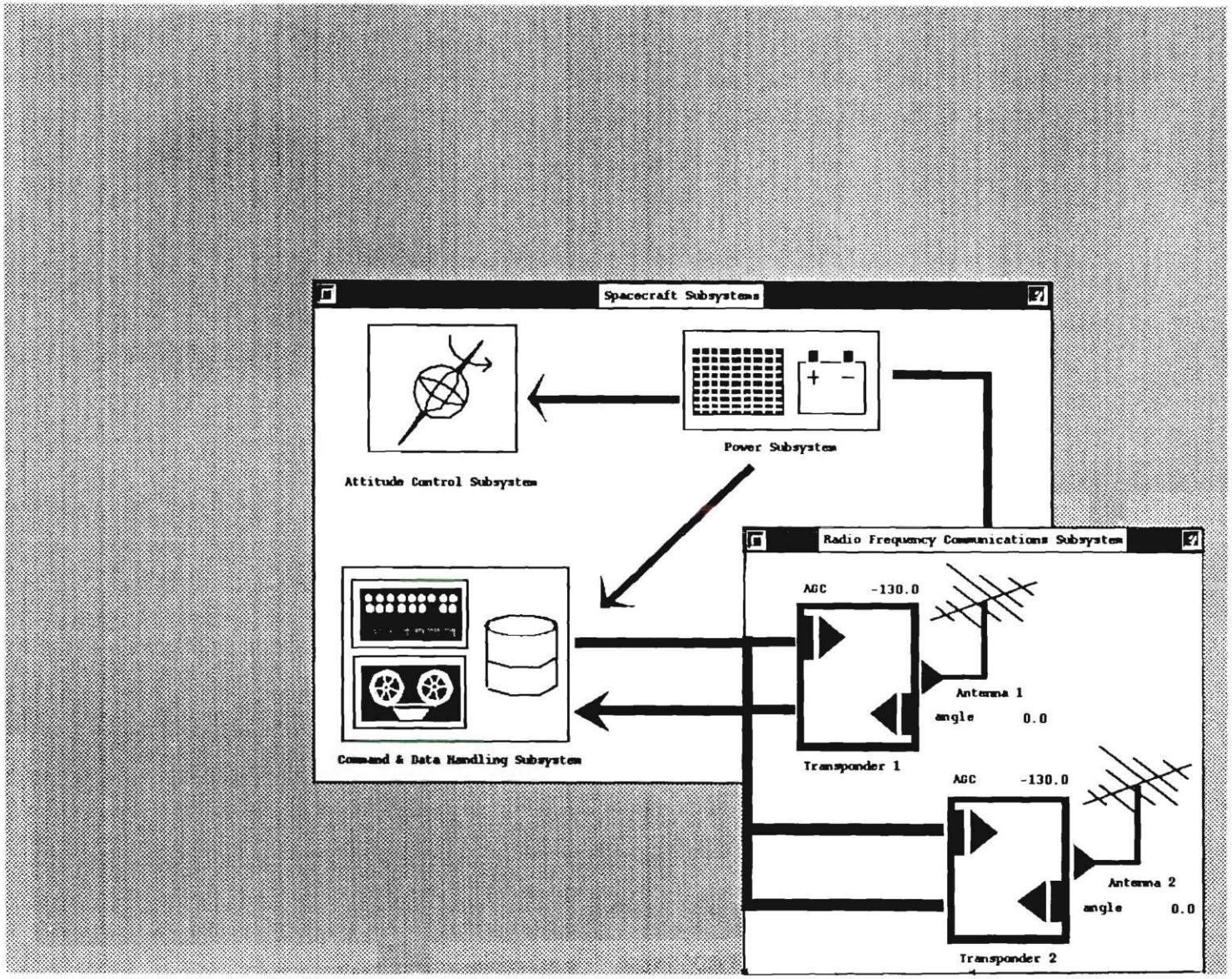


Figure 7-15. The radio frequency subsystem graphic display.

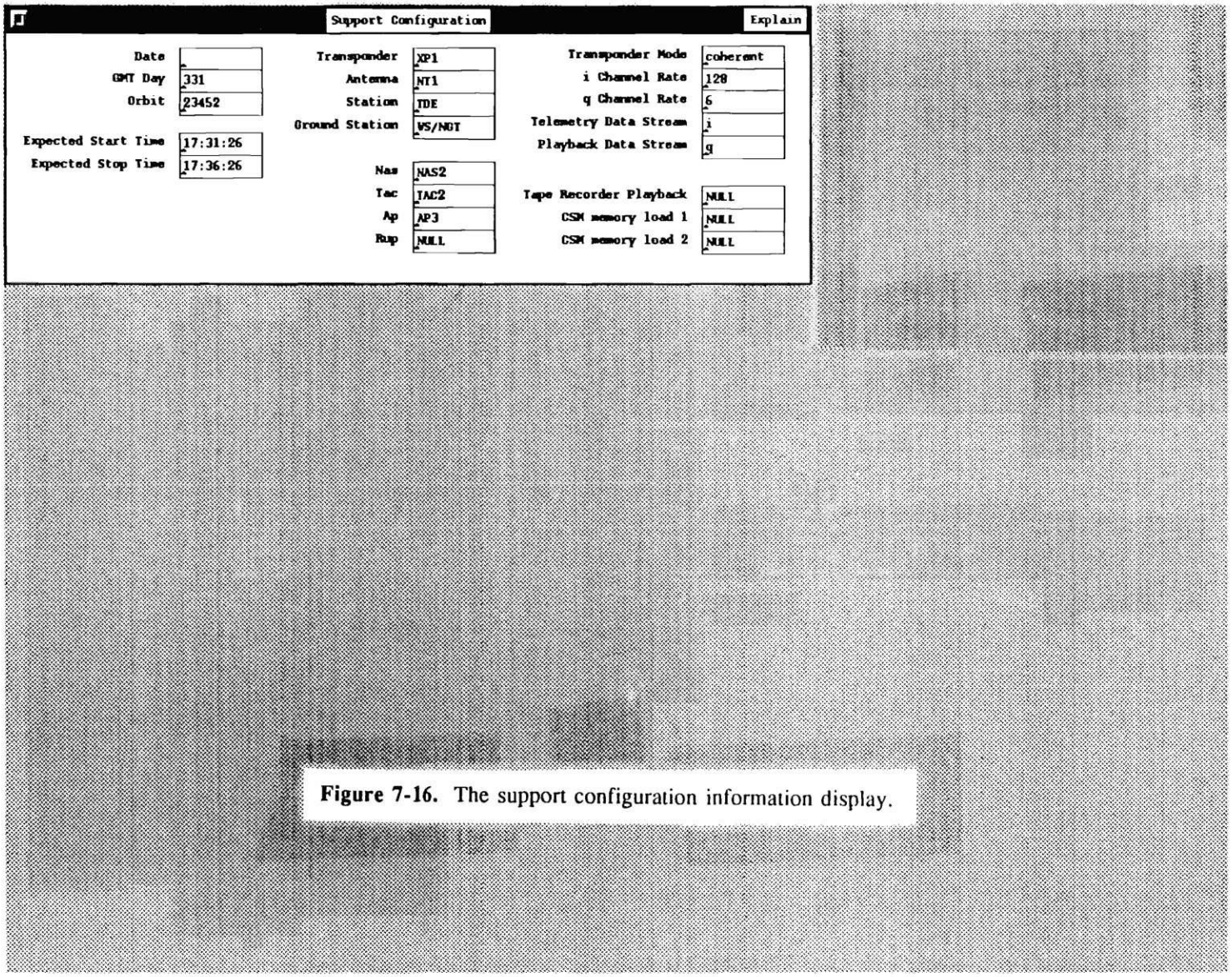


Figure 7-16. The support configuration information display.

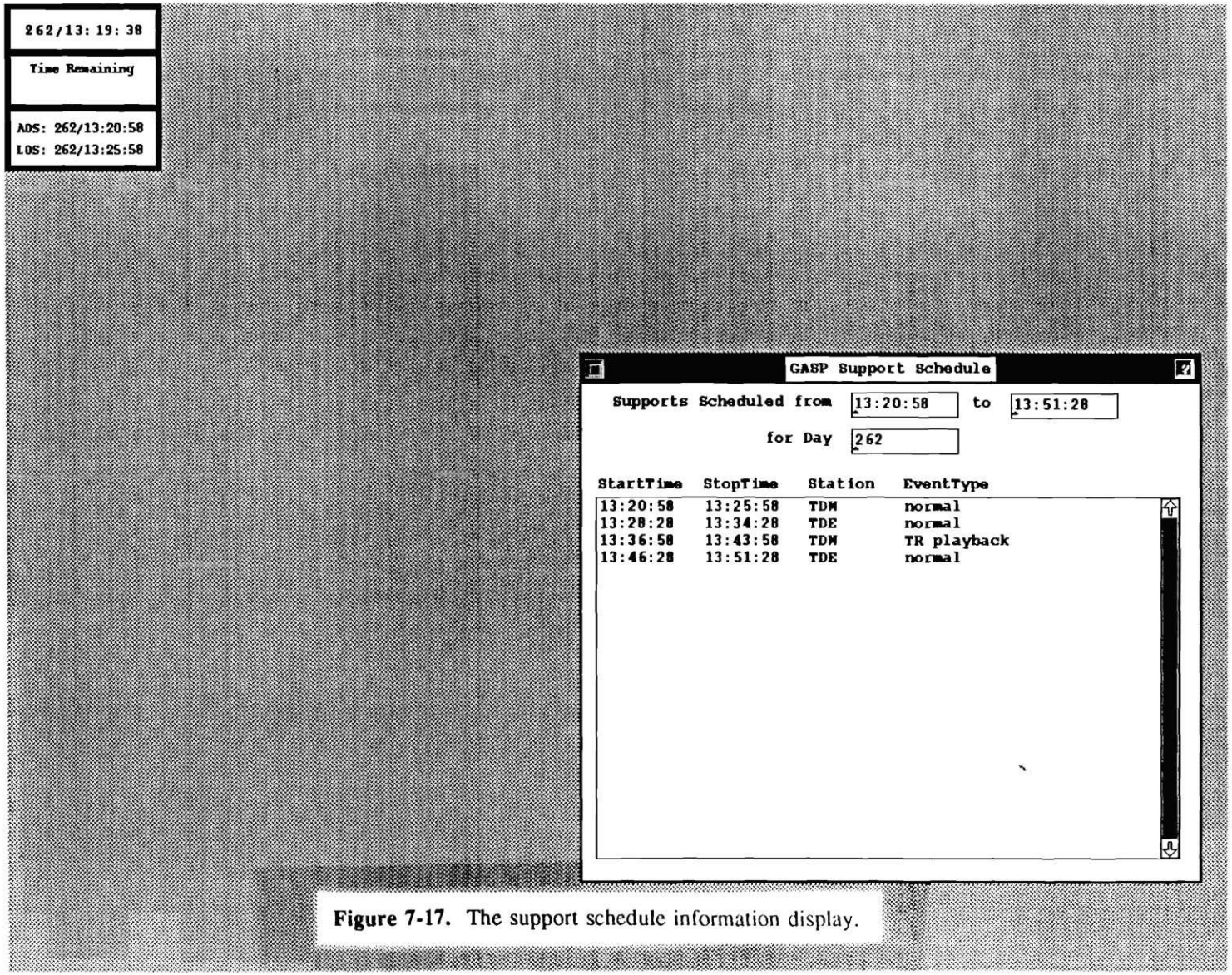


Figure 7-17. The support schedule information display.

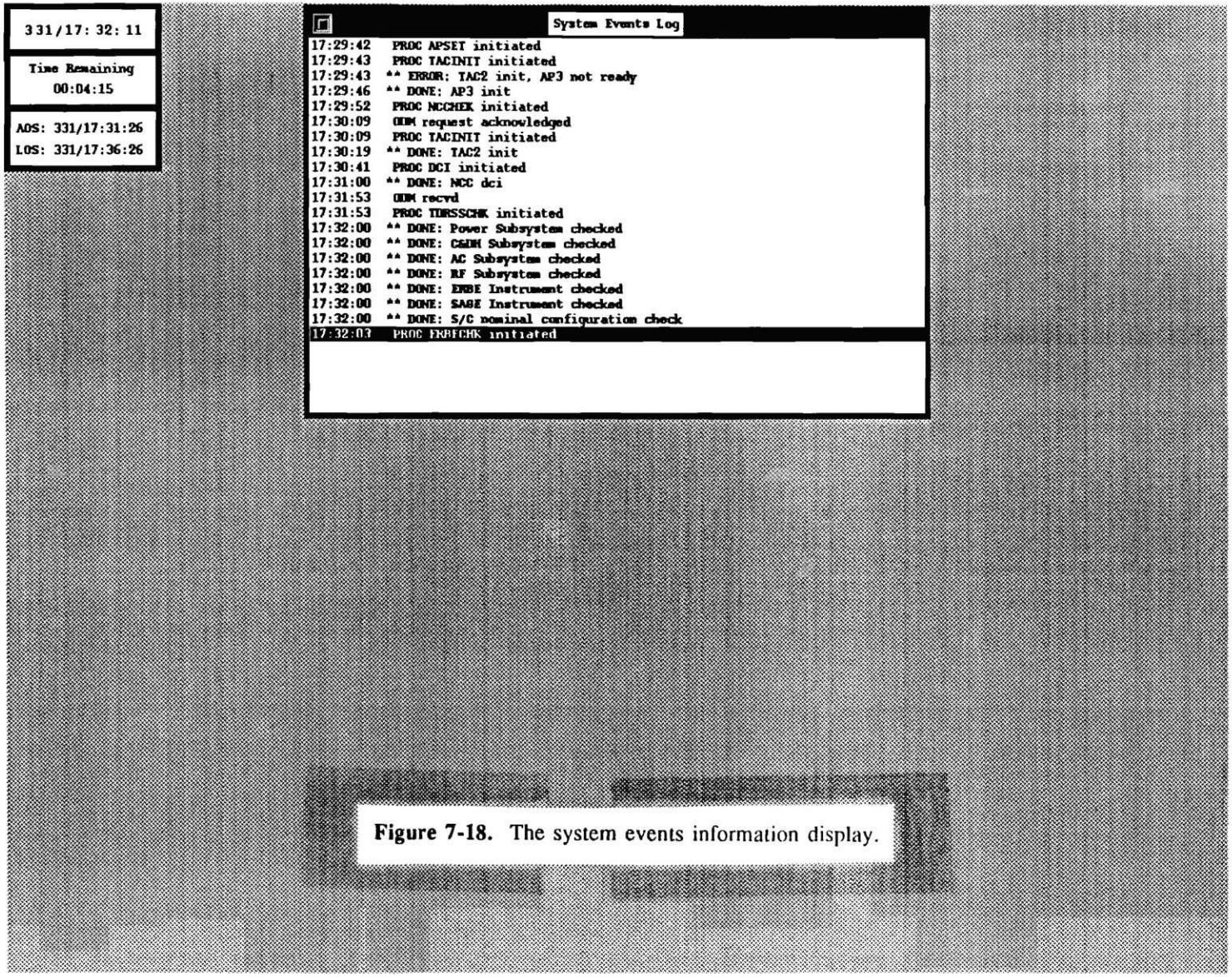


Figure 7-18. The system events information display.

So far, the student's information request actions have been accomplished by the direct manipulation of system objects. There are three other types of information requests that are not represented with objects: support configuration, system events and support schedule. Each information panel is described next.

Support Configuration. The student clicks on a button labelled "support Configuration" to display information about the current support -- expected start and stop times, ground equipment used, for example (Figure 7-16). The information is equivalent to that provided by a pass plan in the task environment (see Figure 6-14).

Support Schedule. The student selects the "Support Schedule" option on a pulldown menu labelled "Status Information" to get this panel (Figure 7-17). This panel is equivalent to the support schedule that can be requested from the pass plan (Figure 6-14).

System Events. The student selects the "System Events" option on a pulldown menu labelled "Status Information" to get this panel (Figure 7-18). This panel is equivalent to the one requested with the button EVENTS on the command panel (Figure 6-8).

Operator Control of System

With respect to operator activities, GT-VITA also displays the various states of some system components and provides the student with the means for controlling these states via a control panel. That is, the student's role as a GT-POCC operator is manifested through the interactive and controllable graphic objects. Each controllable object and its associated control panel is described next. Where appropriate, the control actions on each control panel are mapped correspondingly to those on the task interface as discussed in Chapter VI (Refer to Table 6-2), many of which are also button names on the command panel (Figure 6-6). In general, all control panels have a Cancel button and a Okay button that enables the student to revert a selected action or to execute it.

NCC. When the student clicks on the NCC object on the Goddard Space Flight Center panel, a control panel for sending commands to NCC is displayed (Figure 7-19). The student requests NCC for operation data messages (NCCHEK), request and to inhibit doppler compensation (DCI) shortly before a

real-time pass begins. If there is evidence of data lost, the student requests reacquisition of the forward link signal (FWDURR) and/or the return link signal (RTNURR).

Telemetry And Command Computer. The student clicks on the button labelled "Telemetry and Command Computer" above the TAC object on the MSOCC graphics page to display the control panel for TAC (Figure 7-20). The student initializes the TAC before the start of a support (TACINIT), changes its playback rate from high to low (TACLPR) or from low to high (TACHPR) before a tape recorder playback activity is initiated. The current playback rate is also shown.

Application Processor. The student clicks on the button labelled "Application Processor" above the AP object on the MSOCC graphics page to display the control panel for AP (Figure 7-21). The student initializes the application processor (APSET) before the start of a support, and terminates it after the pass is over (APTERM).

Transponder. The student clicks on a transponder object on the Radio Frequency Subsystem panel to display the control panel for the transponder. The panel shows the current status of the transponder's string parameters: power, mode and designated station (Figure 7-22). The panel also shows the status of the transmitter and the receiver. The transmitter or the receiver become active during data transmission. The power, mode and station values are usually planned in advance. However, at times, the transponder mode may be subjected to scheduling errors from other NASA facilities. In that case, the student changes the transponder mode from coherent to non-coherent (e.g., XP1NOCO) or from non-coherent to coherent (e.g., XP1COHO). More importantly, during a real-time support, an experienced operator may be allowed to command state changes of these parameters in response to unanticipated emergencies such as early termination of a support or sudden station failure. Since unanticipated events can not be modeled, the goal of GT-VITA is to train the student to cope with problems that can be anticipated, and to show the student the resources available to cope with those that cannot.

Antenna. The student clicks on an antenna object on the Radio Frequency Subsystem panel to display the control panel for the antenna (Figure 7-23). Similar to the transponder, the control panel is useful for trouble management. That is, the panel shows the station that the antenna is making contact with. The choice of station has been planned in advance. However, in case of emergencies, the operator

may redirect the antenna to another available station. For GT-VITA, the student does not exercise this control panel, but the student becomes aware of the available option that may be needed to manage unanticipated problems during online operations.

Tape Recorder. The student clicks on a tape recorder object on the Command & Data Handling Subsystem panel to display the control panel for the tape recorder. This panel displays the current states of the tape recorder in terms of its string parameters: power, rate and mode (Figure 7-24). The panel also shows the various state values that each parameter can take on. This control panel helps the student perform an FOT analyst's duty to ensure proper scientific data collection and timely data playback. The student commands the tape recorder to change the mode to standby (e.g., TR1STBY), playback (e.g., TR1PBK), fastforward (TR1FF) or record (e.g., TR1 REC). The student commands the tape recorder to change its rate of recording or playback from low to high (e.g., TR1HI), or from high to low (e.g., TR1LOW). The tape recorder should never need to be turned off, but in rare cases, the operator can do so to rectify unexpected problems.

Command Storage Memory. The student clicks on the command storage object on the Data & Handling Subsystem panel to obtain a view of the memory partitions and to exercise control actions on the command storage memory (see Figure 7-14). The control panel portion of the Command Storage Memory panel is boxed in dotted lines. This panel also shows the current status of the command storage memory in terms of its mode and pending memory loads to be uplinked from the ground. This control panel helps the student perform an FOT analyst's duty to ensure timely transmission of memory loads from the ground and to verify the spacecraft memory contents periodically. The student disables execution of the command storage memory (CSM1DIS), initiates the load process (CSM1LD), specifies the load (e.g., LD-123A), "dumps" the memory contents to the ground (CSM1DMP) or enables the command storage memory.

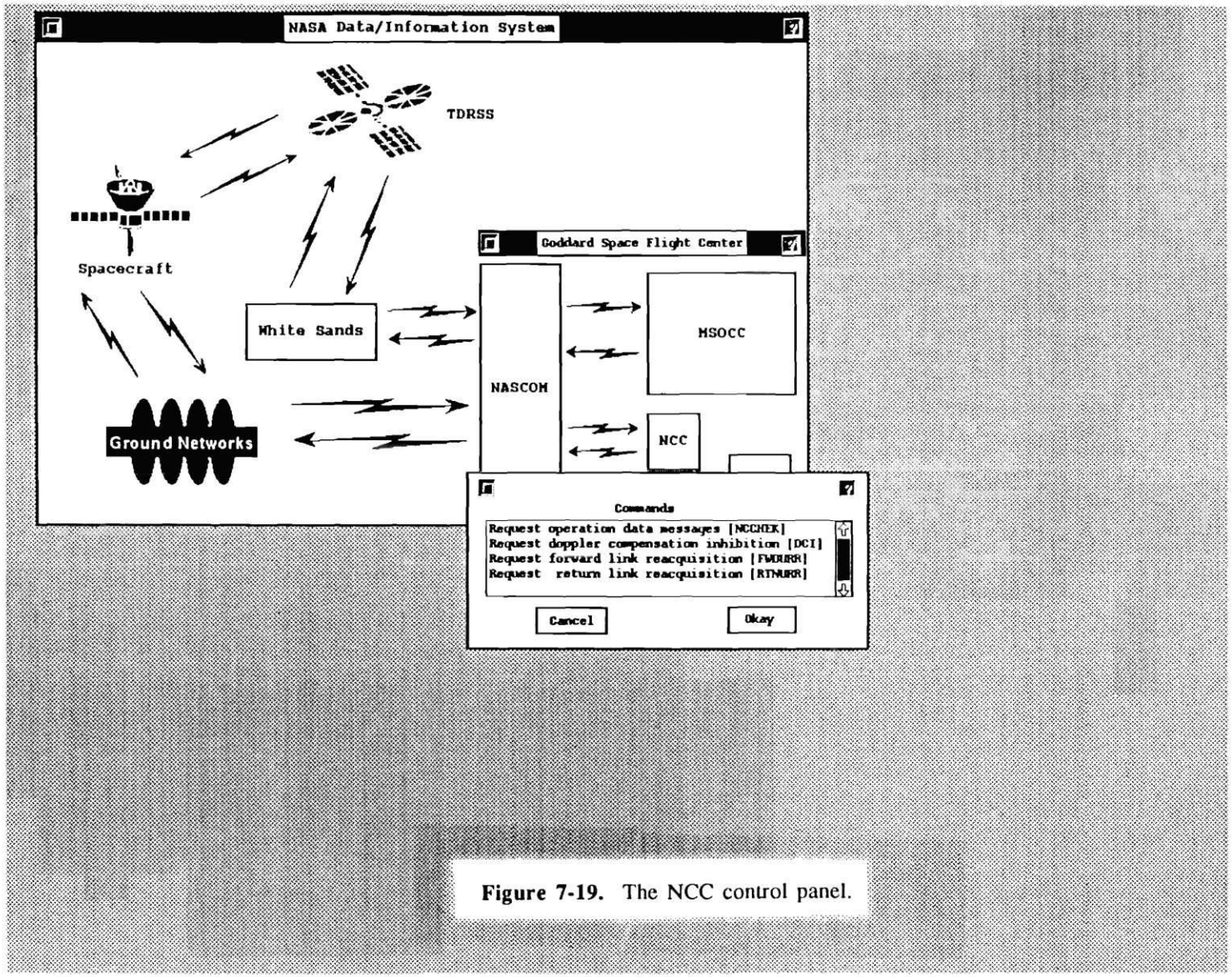


Figure 7-19. The NCC control panel.

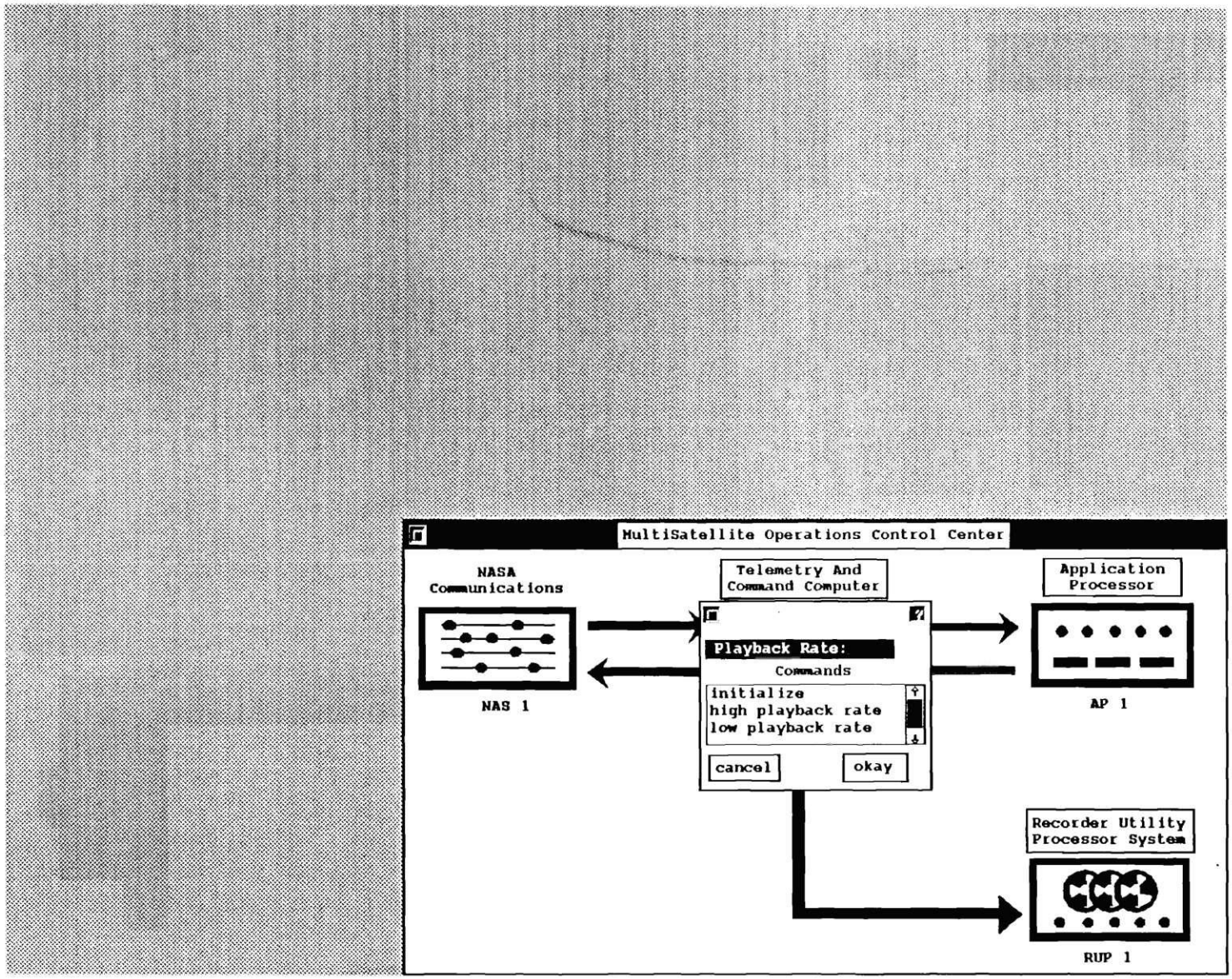


Figure 7-20. The Telemetry and Command Computer (TAC) control panel.

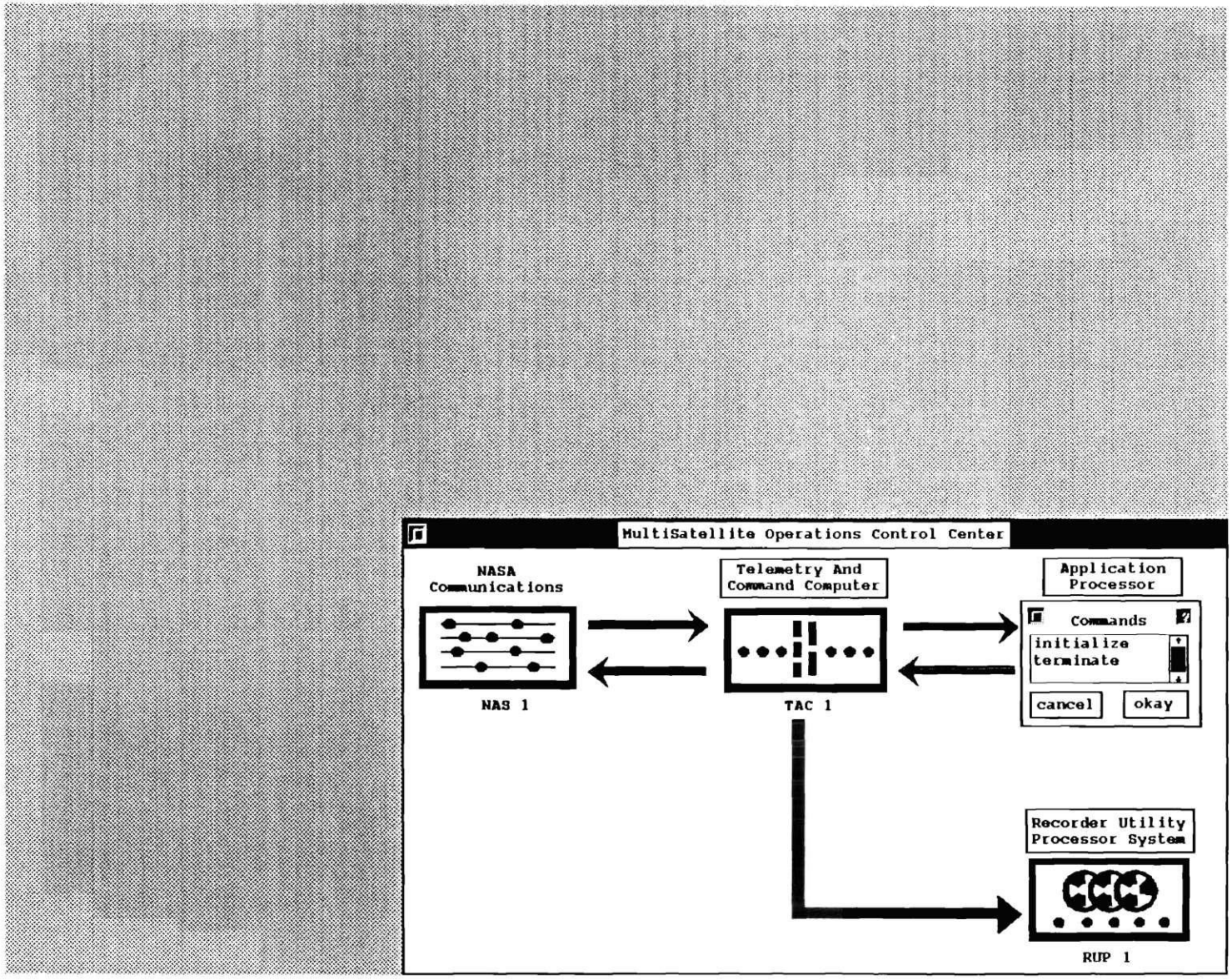


Figure 7-21. The Application Processor control panel.

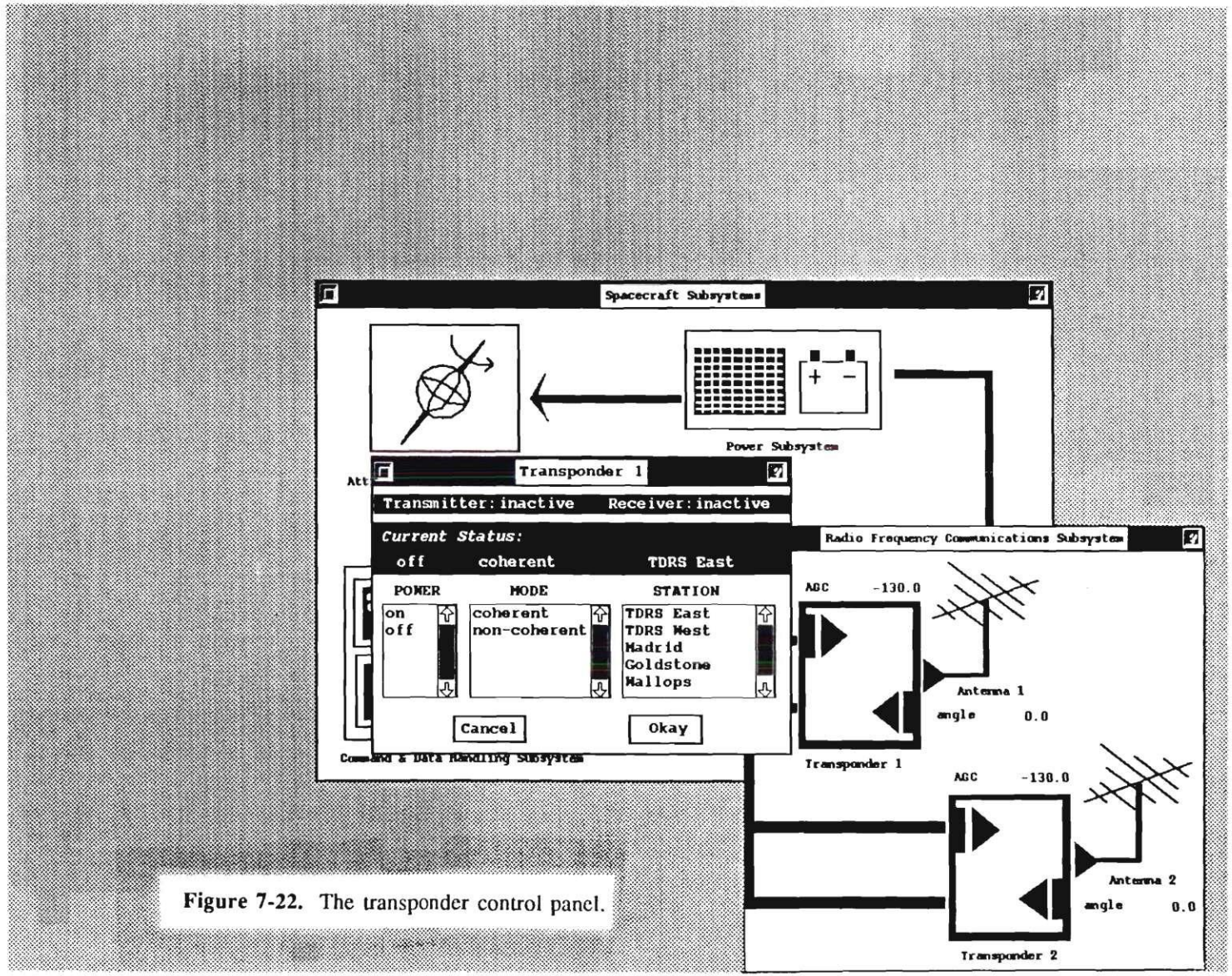


Figure 7-22. The transponder control panel.

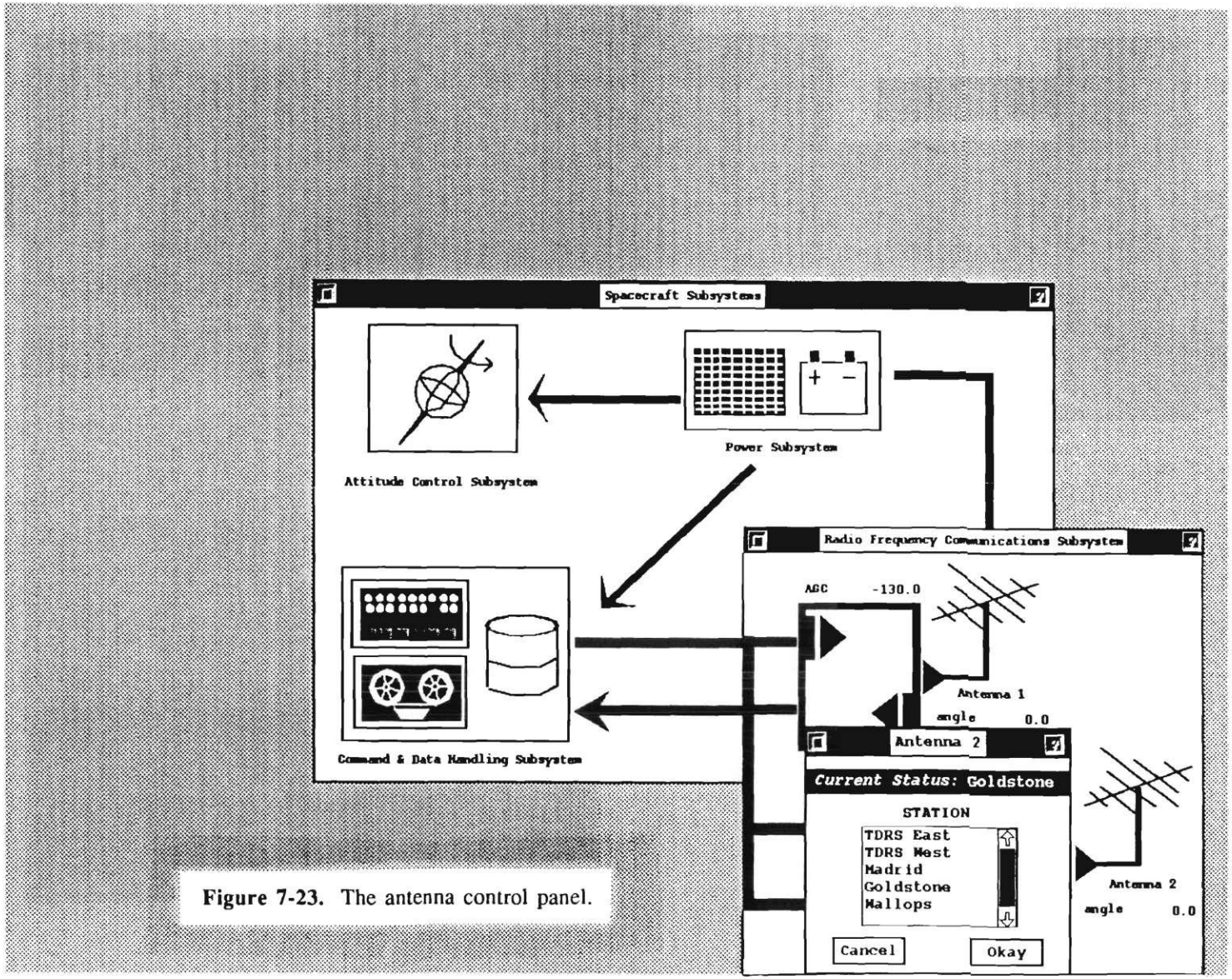


Figure 7-23. The antenna control panel.

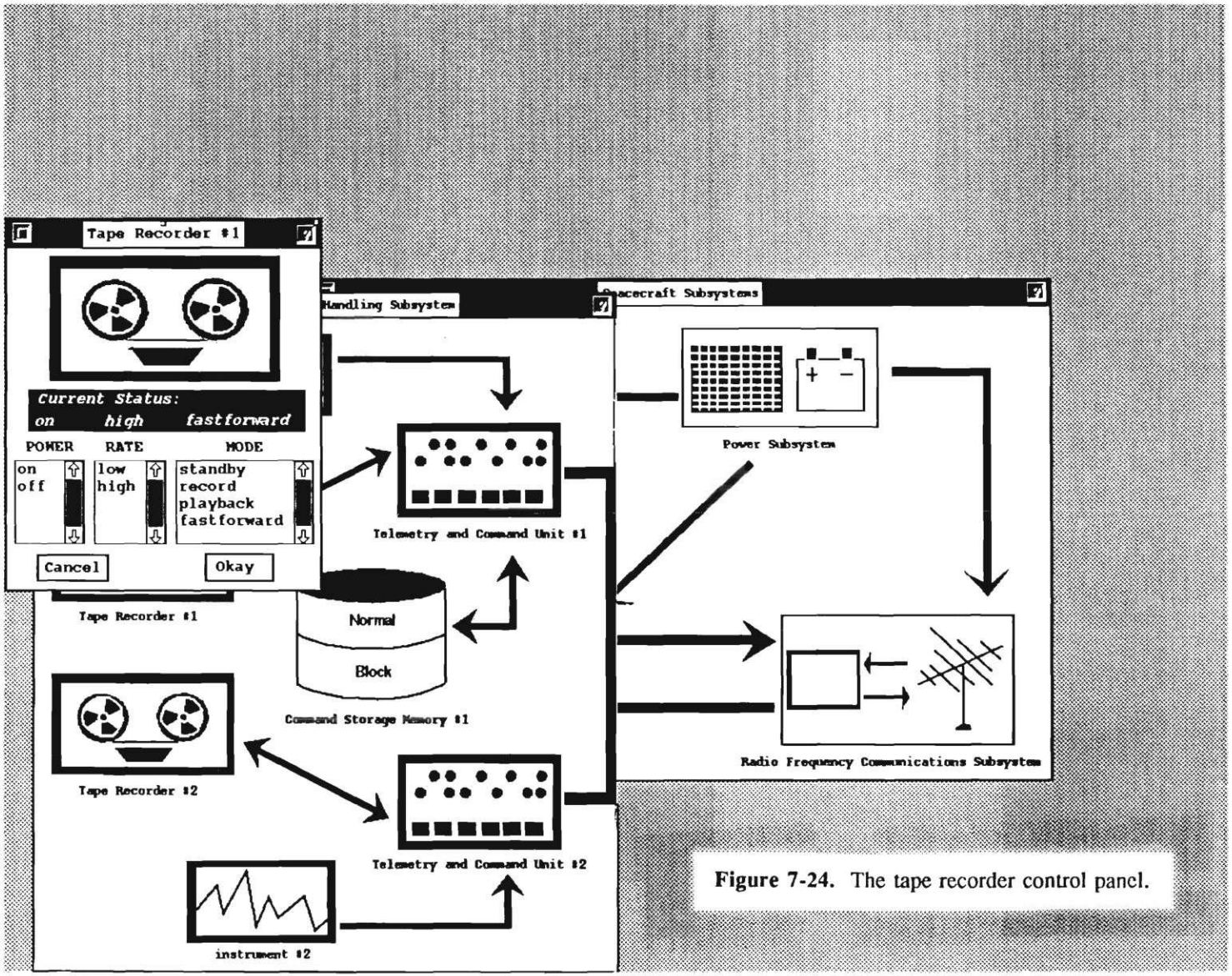


Figure 7-24. The tape recorder control panel.

So far, the student's system control actions have been accomplished by the direct manipulation of system objects. However, there are still four areas of control activities that are not conveniently represented with system components: pre-pass verification, normal bookkeeping and event recording, communication with other facilities, and monitoring spacecraft with STOL procedures. GT-VITA provides a pull-down menu of control panel requests for these four groups of "Other Actions" (Figure 7-25).

Verification. The student selects the "Pre-pass Verification" option on the pull-down menu to display the Verification panel (Figure 7-26). Before the start of a support, the student verifies the application software (SOFTWARE), checks the upcoming support against the scheduled list of supports from NCC (SCHEDULE), and verifies the latest contents listing of the spacecraft's command storage memory (CSMVLD). These three actions are equivalent to the ones shown on a pass plan in the task interface (see Figure 6-14). The mneumonics for the verification commands represent the GT-VITA's abbreviated name for the action (i.e., action node name), as defined in the operator function model (see Table 6-2).

STOL Procedures. The student selects the "STOL Procedures" option to display a selection of procedures shown in Figure 7-27. During real-time supports, the student executes status checks on spacecraft subsystems (TDRSSCHK) and the science instruments (ERBECHK and SAGECHK), that are written in STOL commands. When a support requires a command storage memory load activity, the student executes the VLDVLD directive to request validation output for the command storage memory. All these STOL procedures correspond to those found on the command panel of the task interface.

Bookkeeping Actions. The student selects the "Bookkeeping" option to display the control commands for four bookkeeping tasks: Pass Plan Completion (XPLAN), Tape Recorder Data Accountability (DATA), Events Report Completion (XEVENT), and Anomaly Report Completion (XANOM) (Figure 7-28). The mneumonics here correspond to the task node names for the bookkeeping function (see Table 6-2). For each bookkeeping task, the student selects from a list of actions whose abbreviations also correspond to the action node names in Table 6-2. All bookkeeping actions are symbolic: the student learns about the *concept* of requesting a page and filling it in, and not actually how to do it. In this way, while training on the system interfaces provided by GT-VITA, the student does not need

to worry about the details of each bookkeeping pages in the task interface as yet (see Figures 6-14 to 6-18). To fill in an event or anomaly report, the student needs to specify the system component that is causing the problem. To do so, the student chooses from the pull-down menu labelled "Details" to select the appropriate component.

Communication. The student selects the "Communication" option to display the facilities that a FOT analyst must contact in case of problems (Figure 7-29). This panel is equivalent to the one in the task interface (Figure 6-12) that represents GT-POCC's version of the voice-link communication. When communicating with DOCS, the student is required to specify the object of communication by selecting the "Details" pull-down menu.

In short, the GT-VITA system interface not only serves as a visualization tool, but also provides the means for the student to learn about and exercise the duties of a FOT analyst. However, the power of GT-VITA lies in its pedagogy that structures how a novice learns about the system with the system interfaces provided by GT-VITA.

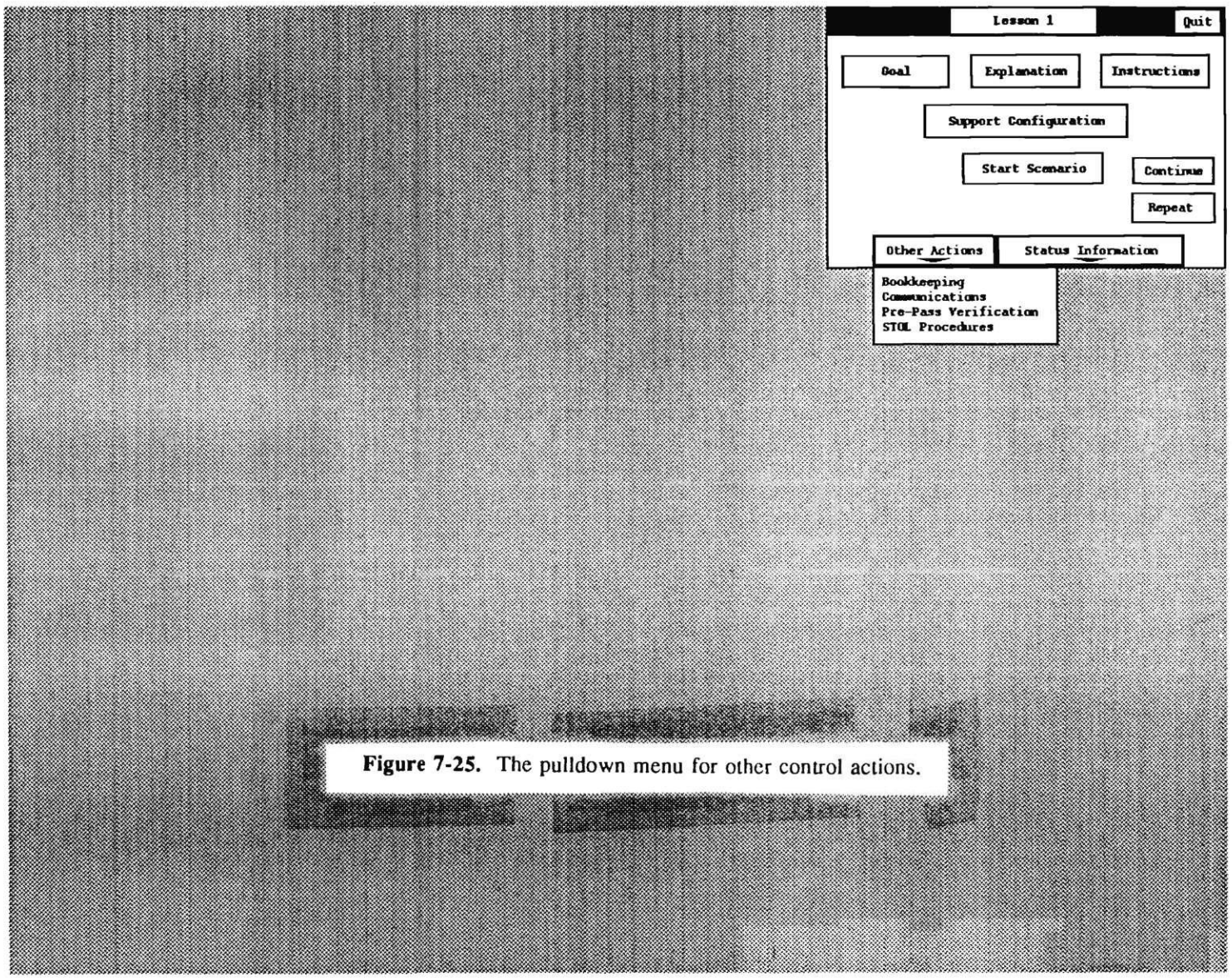


Figure 7-25. The pulldown menu for other control actions.

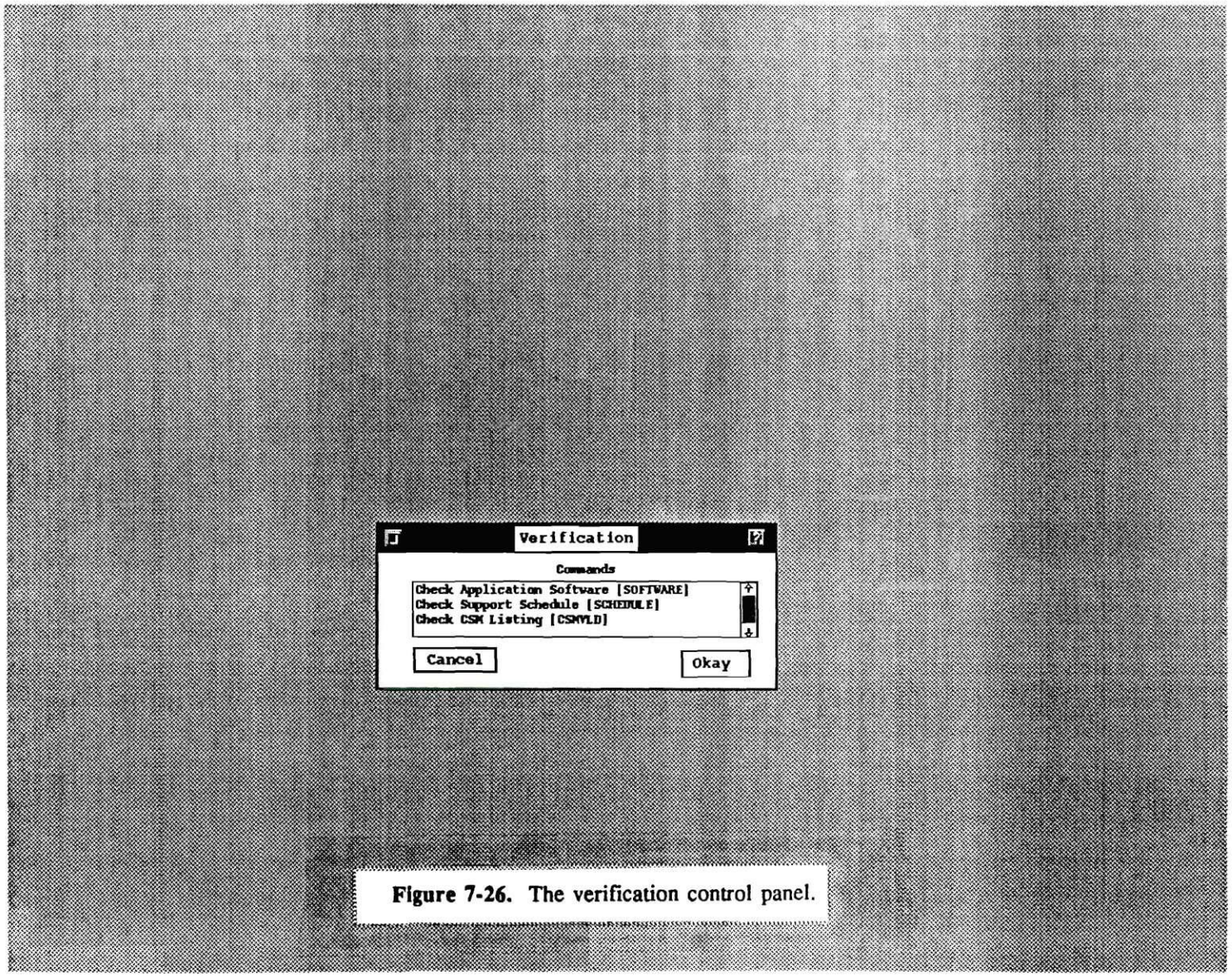


Figure 7-26. The verification control panel.

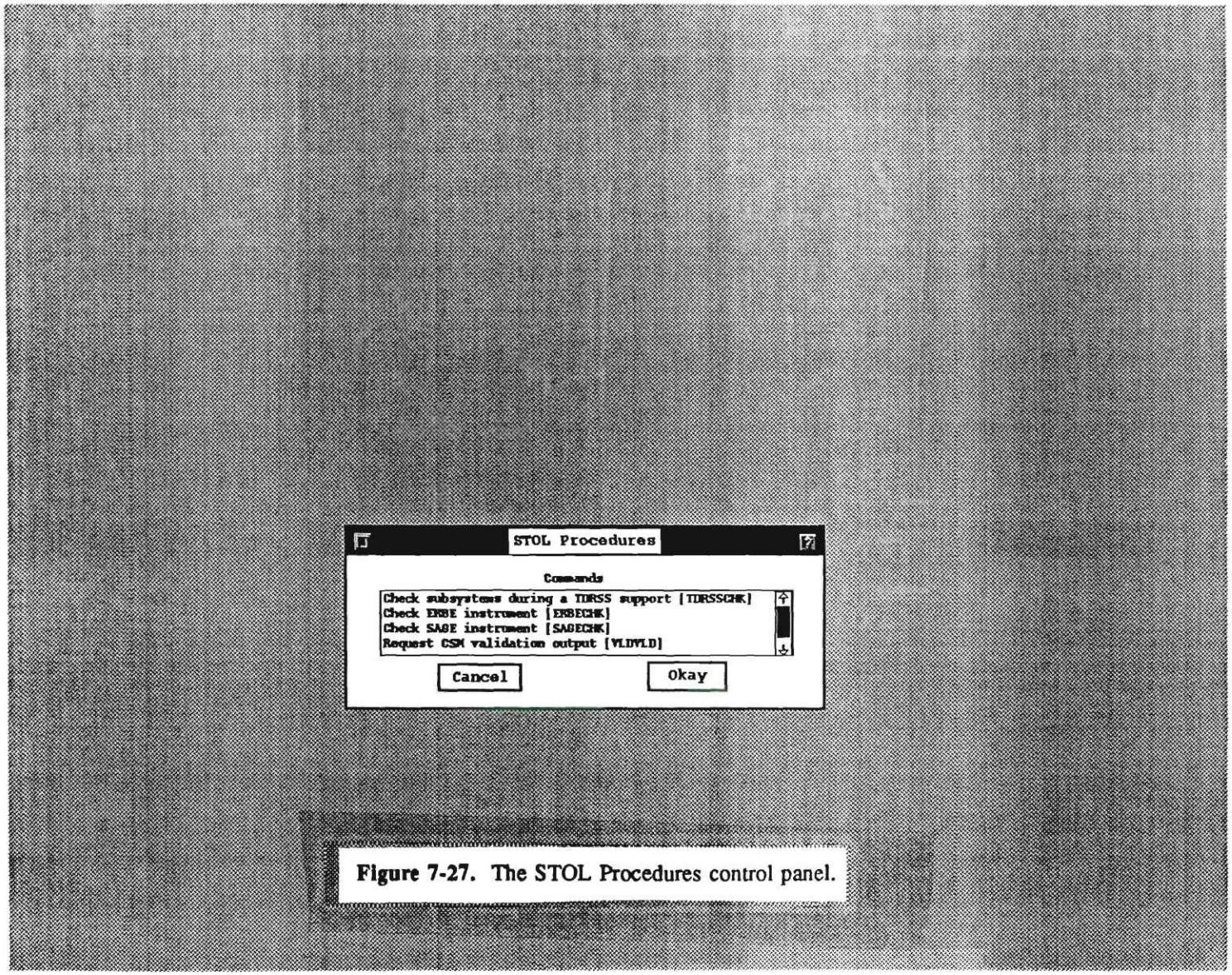


Figure 7-27. The STOL Procedures control panel.

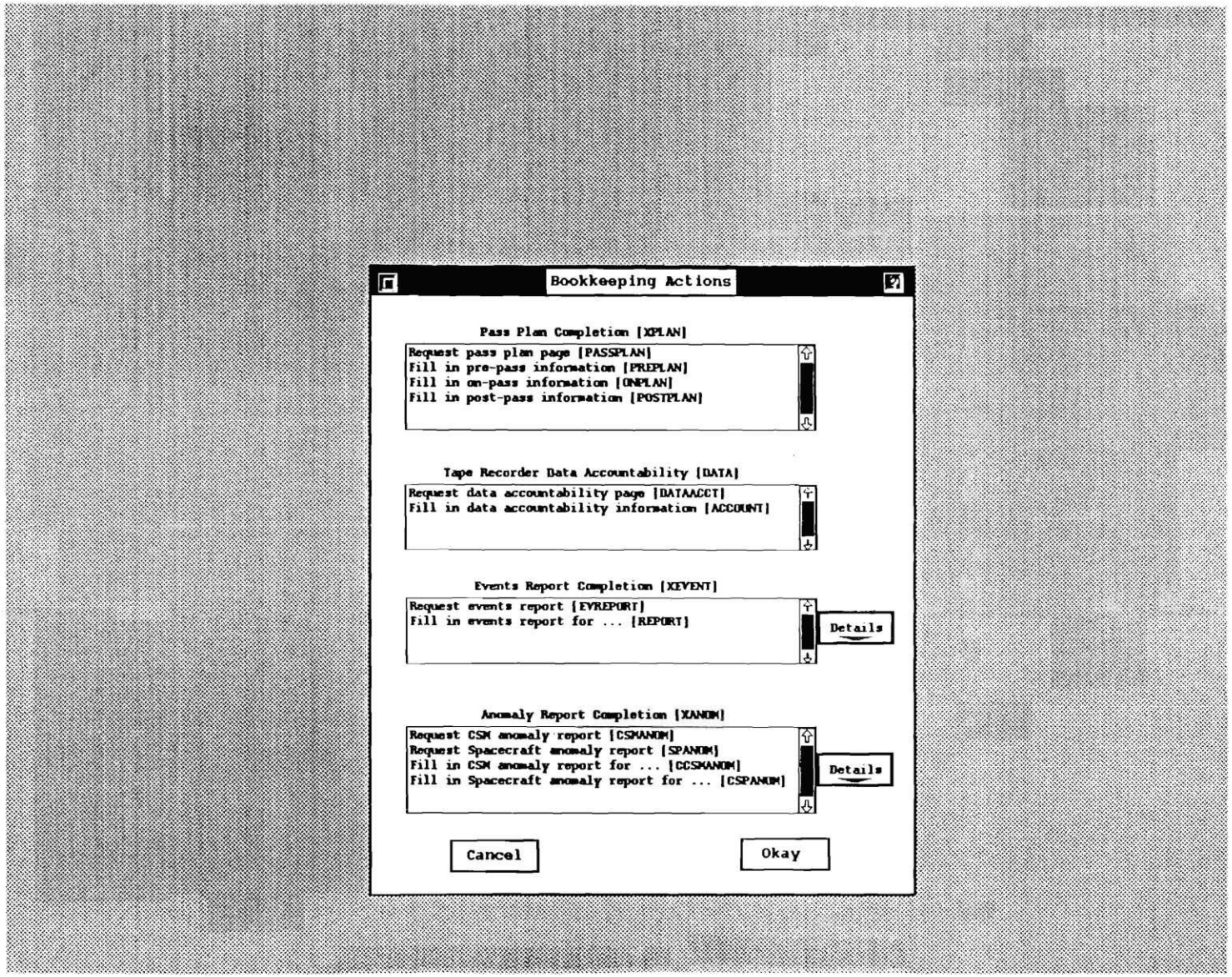


Figure 7-28. The bookkeeping actions control panel.

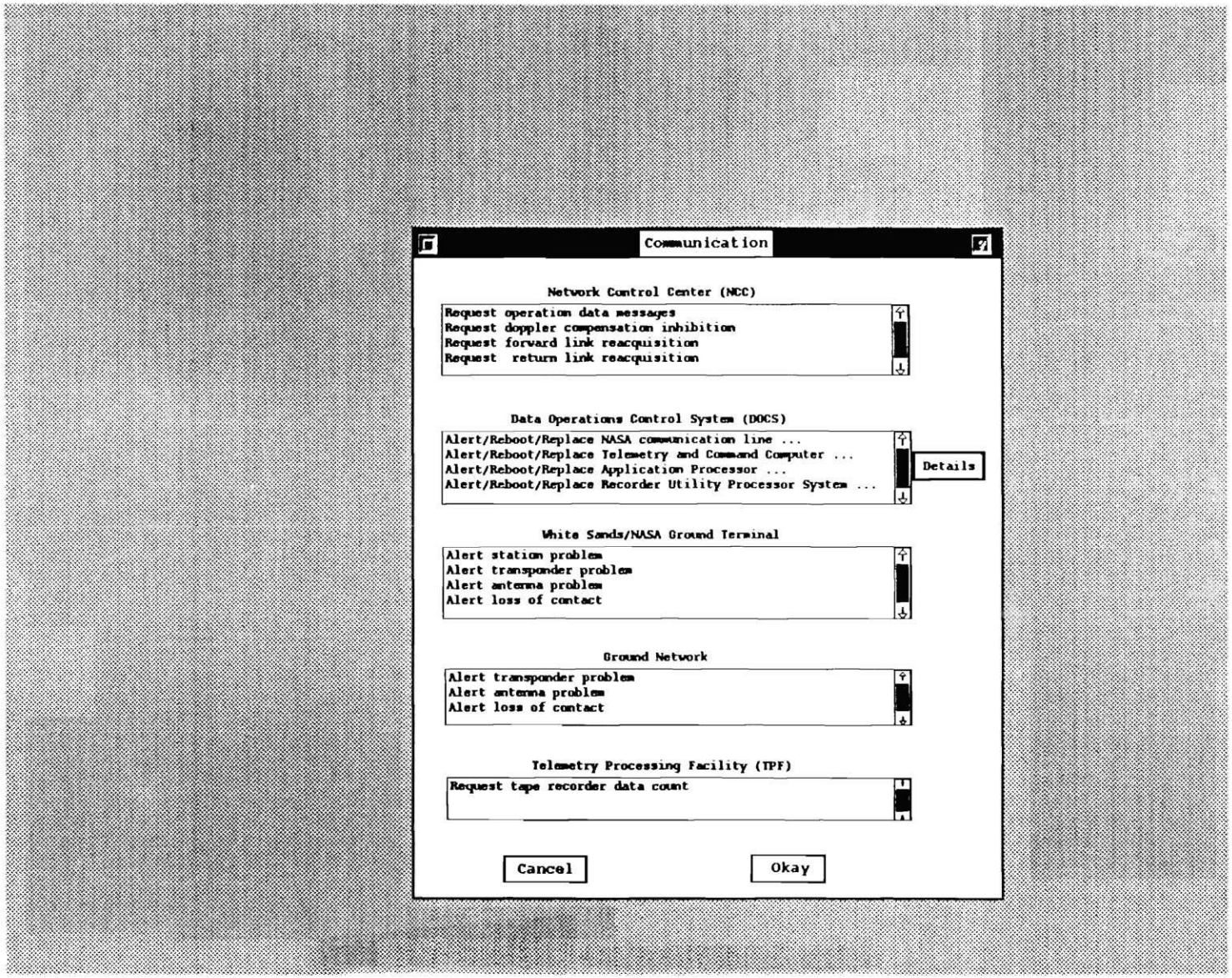


Figure 7-29. The communications control panel.

GT-VITA Tutorial Interfaces

In order to systematically teach the student about the NASA system and the control activities of an operator, GT-VITA structures its pedagogy as lessons. The interaction between the tutor and the student is governed by the instructional goals for a particular lesson. This interaction is carried out with tutorial dialog interfaces. Besides a means for instruction, these interfaces provide the student with feedback about the student's interaction with the domain system interfaces.

GT-VITA is developed using the pedagogical design proposed in Chapter IV. Seven out of the eight lesson types (see Table 4-1) are implemented in GT-VITA. Each lesson type has a set of tutorial interfaces dedicated to the instructional strategies for achieving the lesson type's instruction goals. There is also a set of tutorial interfaces that are shared between lesson types. In either case, all lesson types exhibit many common features that enable the student to easily transition between lesson types during training. This section describes GT-VITA's general features, illustrated by representative tutorial displays. The functionalities of each lesson type are detailed in a walkthrough presented in the next section.

Figure 7-30 shows an example of the GT-VITA interface at the start of every lesson. On the top left corner of the left monitor are displays of a digital clock, the remaining seconds in a real-time support, and the expected acquisition of signal (AOS) and loss of signal (LOS) for the current support. The top right corner of the right monitor shows a main panel for a lesson called the lesson panel. All tutorial interactions are initiated from the lesson panel. This initial interface configuration (the lesson panel plus the timing panels) is preserved throughout the entire lesson.

The Lesson Panel

Every lesson type has a lesson panel that contains several rows of labeled buttons that the student clicks on. Figure 7-31 is an example of a lesson panel. Generally, for the lesson to begin, the student must query each button from left to right, and on every row from top to bottom, except for the buttons labeled "Continue", "Repeat" and "Quit". Thus, the student clicks on the "Goal" button to get a panel that displays a description of the current lesson's goal (Figure 7-32). For the example in Figure 7-31, the

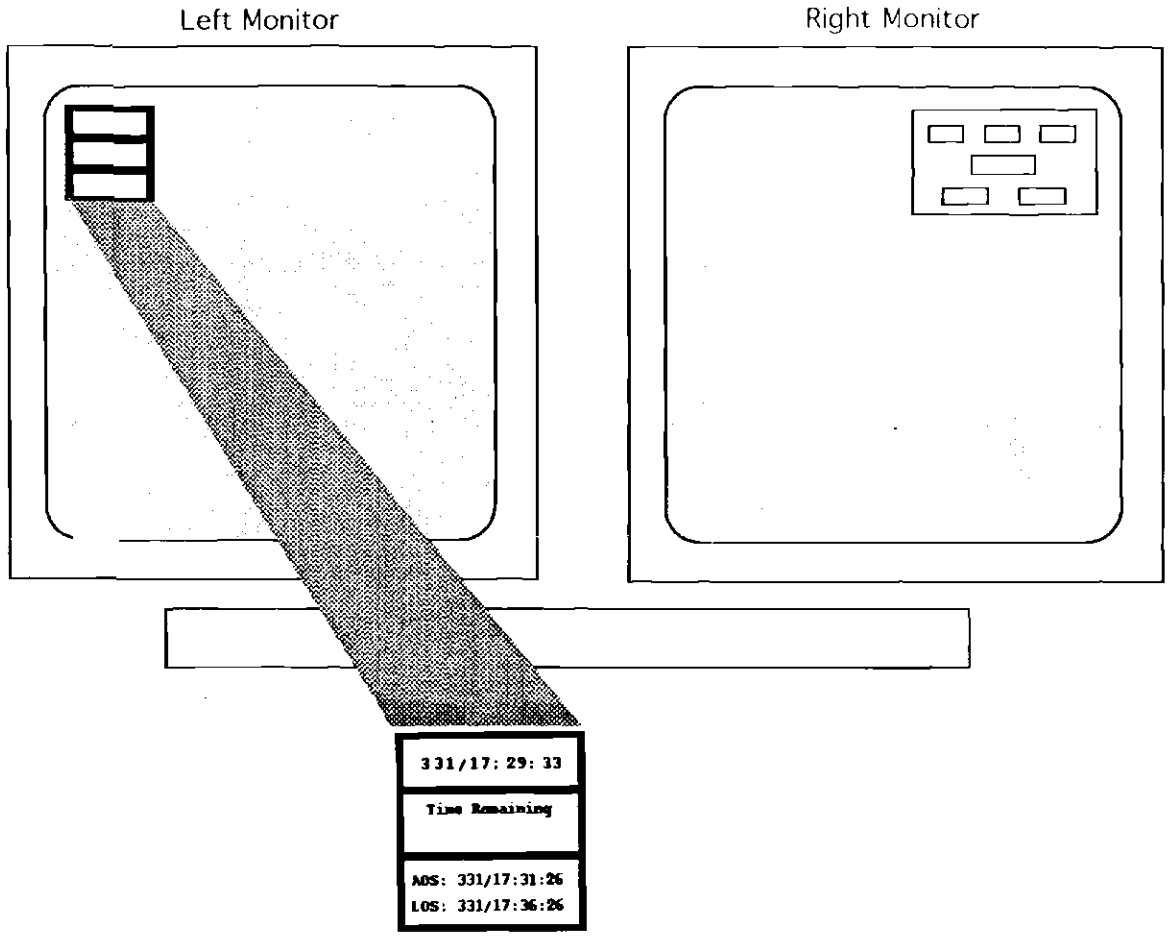


Figure 7-30. GT-VITA's initial tutorial interface.

student next clicks on the "Explanation" button to display more information about the lesson's goal or the contents of the lesson material (Figure 7-33). Then, the student clicks on the "Instructions" button to learn about the student-tutor interactions for the lesson (Figure 7-34).

All but one lesson types have "Support Configuration" and "Start Scenario" buttons on the lesson panel. The "Support Configuration" button displays information about a current support in a panel such as Figure 7-16. The "Start Scenario" button displays a panel that explains the dynamics of the lesson in conjunction with the support scenario running in real time (Figure 7-35). For example, the tutor pauses the scenario to instruct the student about a command. The Starting Scenario panel has three control buttons. The student displays the NASA Data/Information System panel by clicking on the "Show NASA Network" button. From this panel all other system interfaces are accessible. The student initiates a scenario by clicking on the "Start" button. During the course of a lesson, the student can also pause a scenario by clicking on the "Pause" button. The scenario resumes when the "Start" button is clicked upon.

At the end of a lesson, the student makes a decision to either repeat the current lesson or continue to the next one. Generally, the "Repeat" button allows the student to practice the current lesson type on a different support scenario. The "Continue" button allows the student to transition to a new lesson type with different instructional goals. If for some reasons the student wishes to exit GT-VITA, the student clicks on the "Quit" button which is positioned in the top-right corner of the lesson panel.

Explanation Panels

The panels in Figures 7-32, 7-33 and 7-34 are typical of most explanation panels that the tutor uses to provide textual descriptions of important concepts. The Lesson's Goal, Lesson's Explanation and Lesson's Instructions panels pertain to lesson specific textual information. Other explanation panels contain declarative information about system objects and operator commands. Figure 7-36 is an example of a domain specific explanation panel.

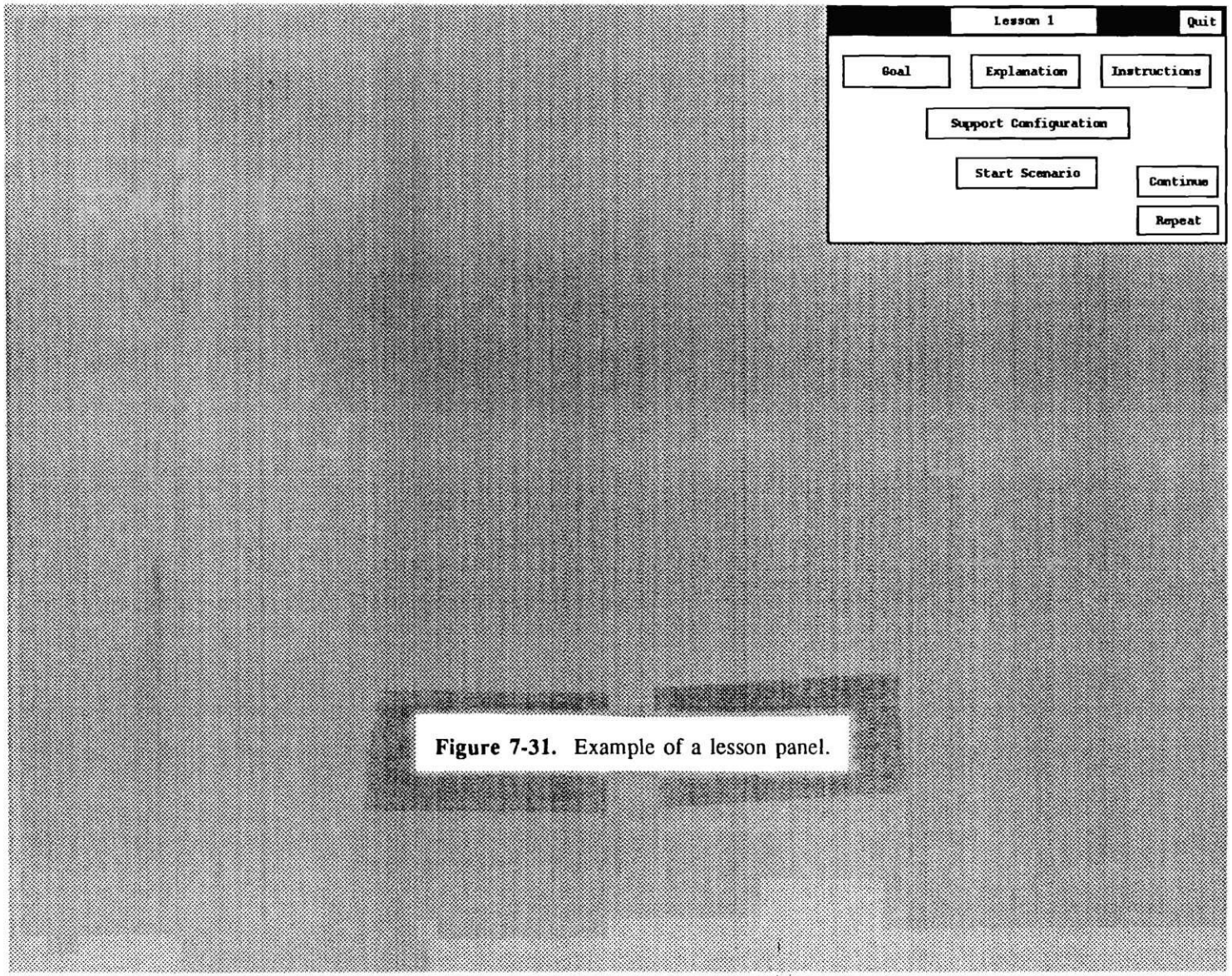


Figure 7-31. Example of a lesson panel.

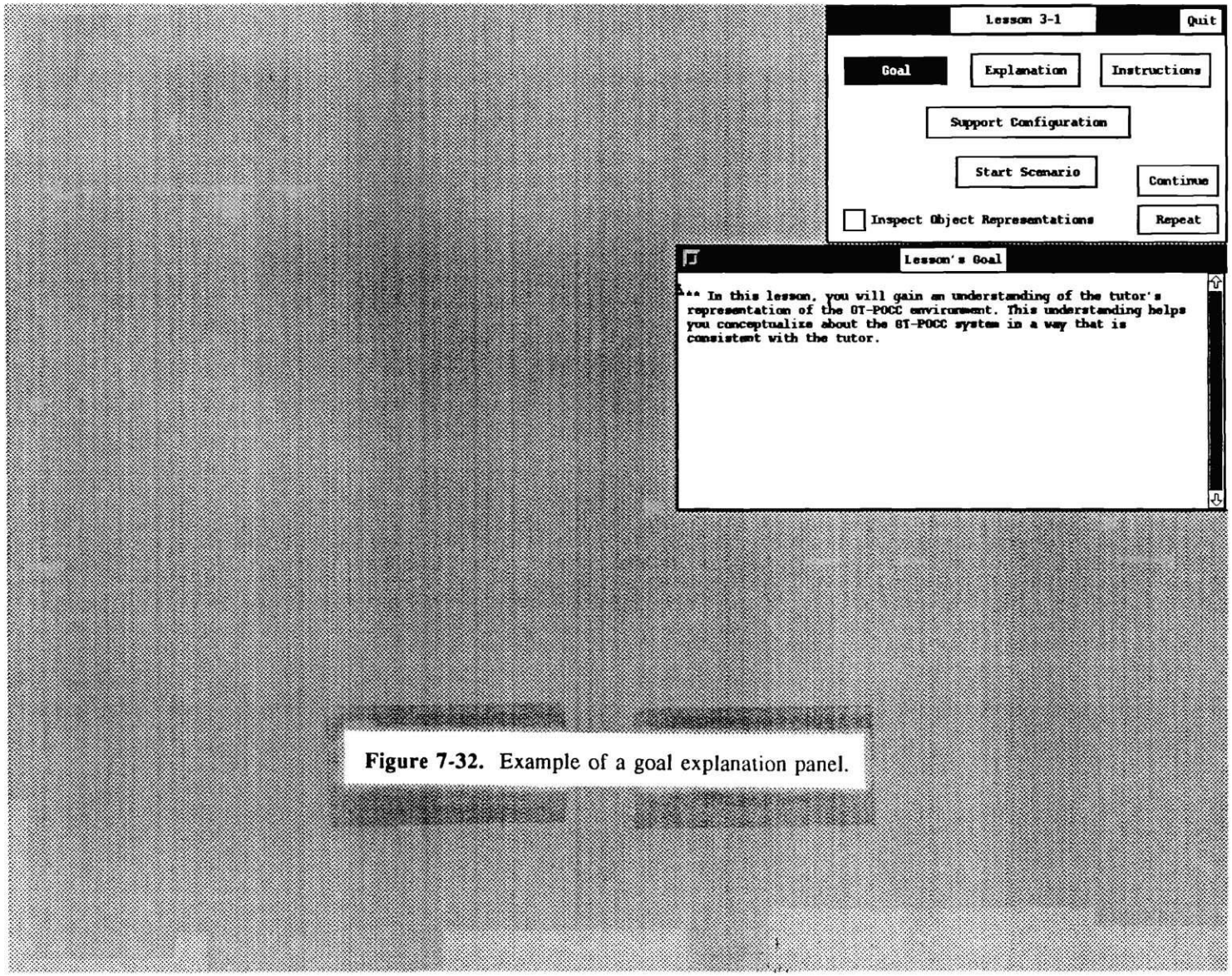


Figure 7-32. Example of a goal explanation panel.

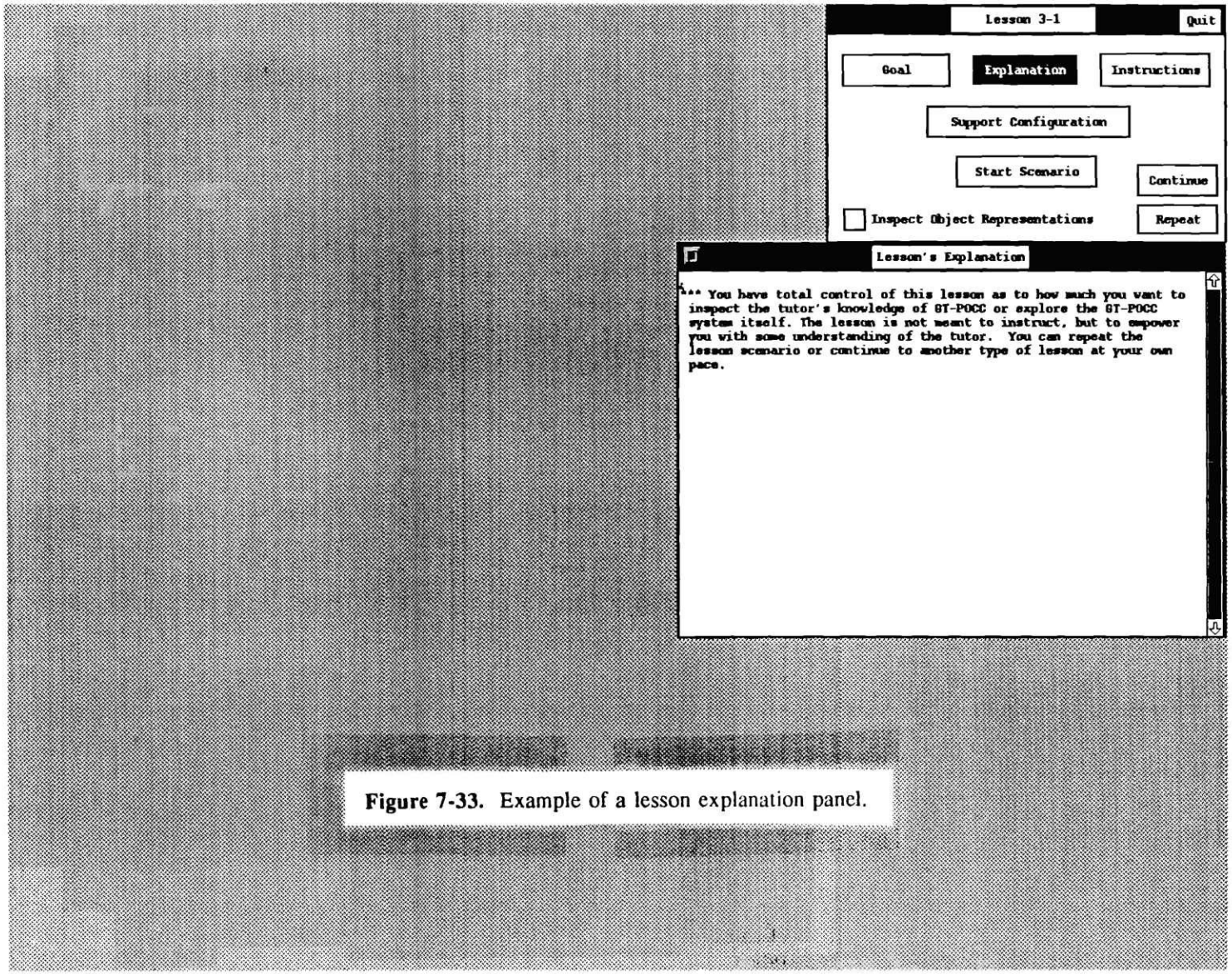


Figure 7-33. Example of a lesson explanation panel.

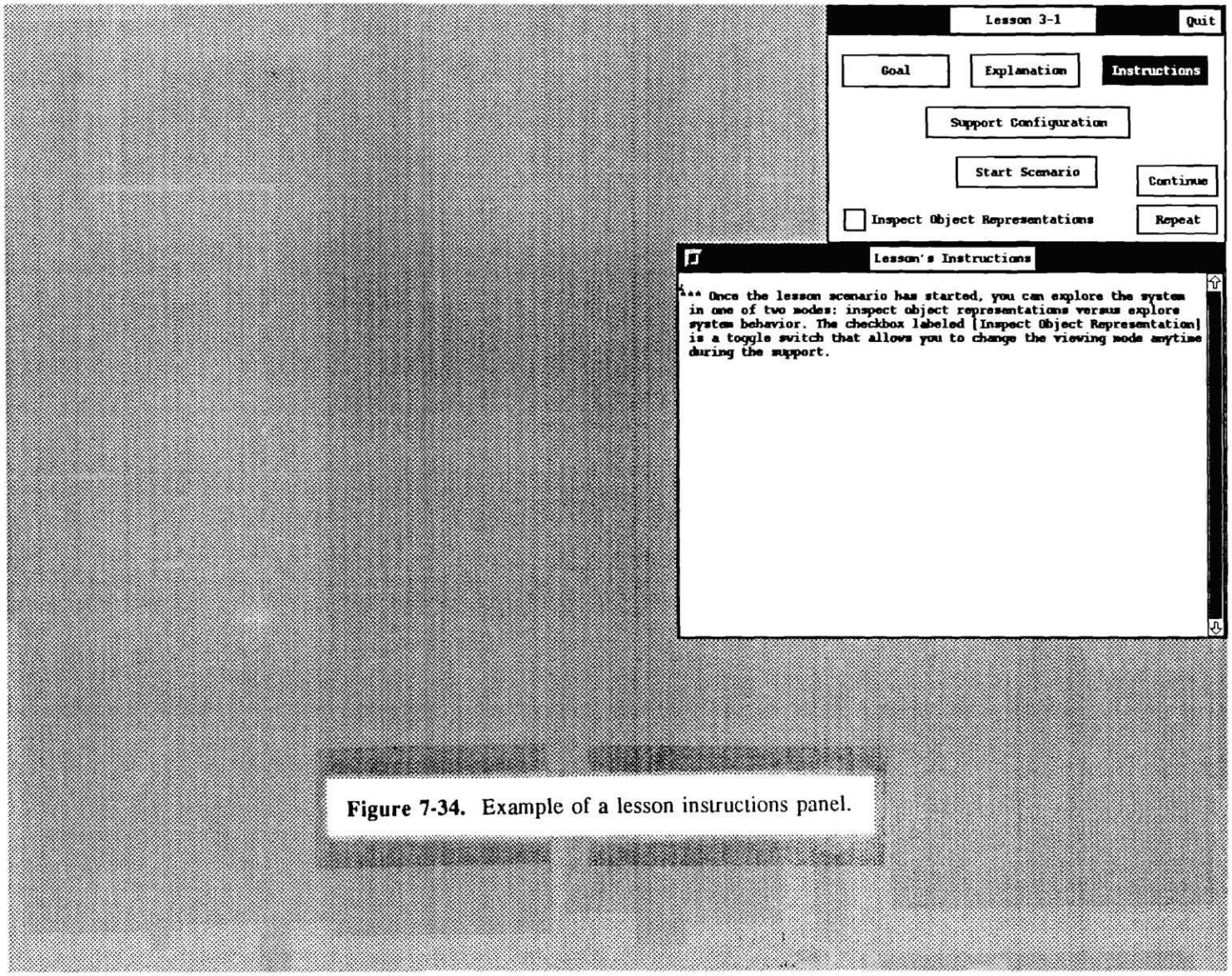


Figure 7-34. Example of a lesson instructions panel.

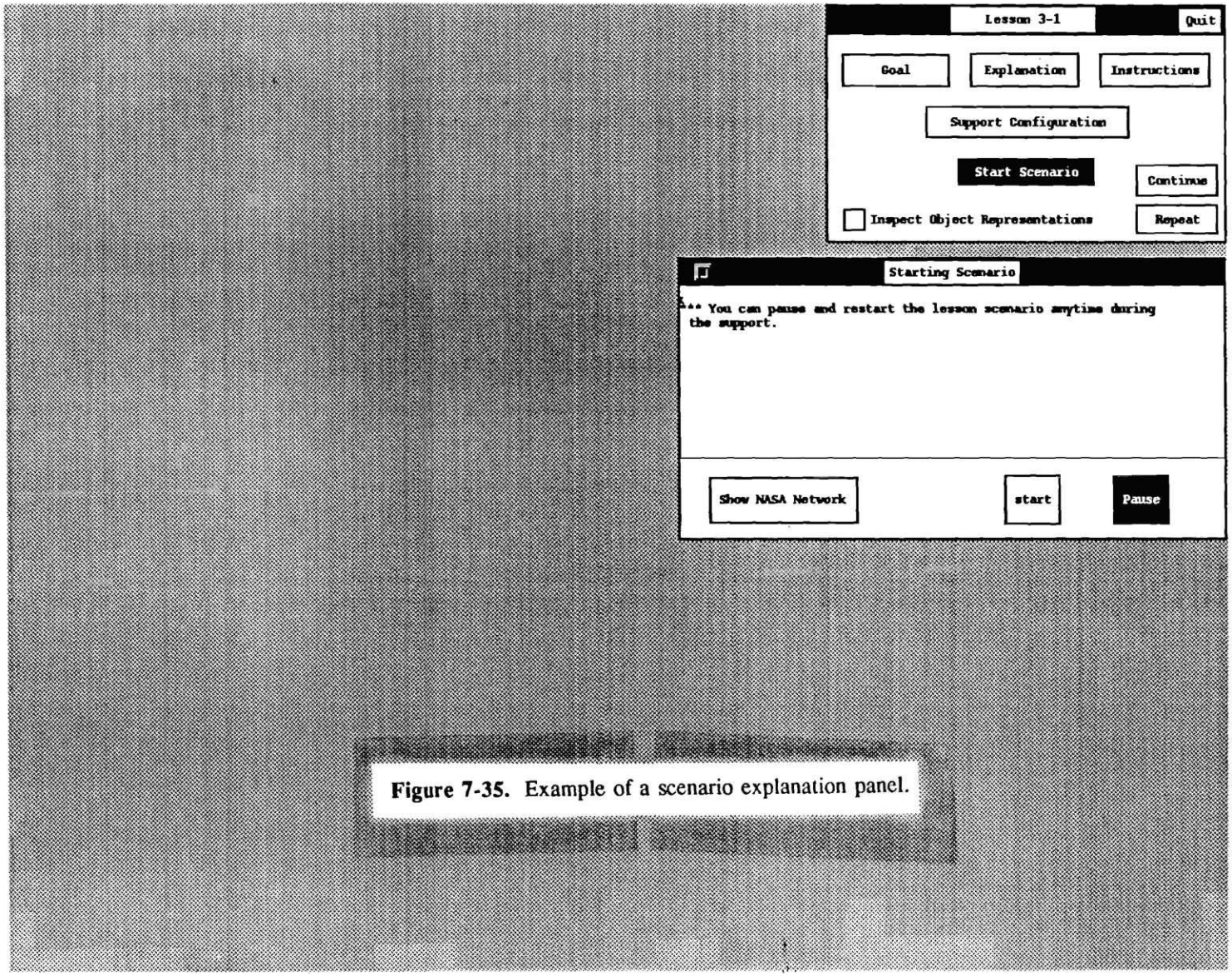


Figure 7-35. Example of a scenario explanation panel.

Information/Feedback Panels

Besides textual descriptions, the tutor also instructs and provides feedback by structuring information in lists such as checklist and actions history. Figure 7-37 shows a list of system failure-related events that the tutor is demonstrating to the student. On most of the information/feedback panels, there is a "Explain" button on the top-right corner of the panel. The student clicks on this button to get a textual description that explains the panel or a selected item on the panel in more details.

The Message Panel

During the course of a lesson, the tutor uses the message panel, accompanied with a beep, to intervene and alert the student to an event. In general, the event can be one of three types. First, there is an error in the student's action, either on the tutorial interfaces or the system interfaces. Second, the tutor is providing instructional information such as when the tutor is ready to demonstrate a concept. Third, the student has successfully completed a task, or a lesson. Figure 7-38 shows three examples of message panels. When the tutor posts a message panel, all system events and student interactions are disabled until the student acknowledges the message by clicking the "Noted" button on the message panel.

Besides acknowledging the message panel, the student is required to close most explanation and information panels to acknowledge that the contents of a panel has been attended to. This is the tutor's way of keeping track of what the student has and has not done. For example, the student cannot click on the "Start Scenario" button on the lesson panel unless all previous rows of buttons have been attended to; that is, the student must know about the lesson's goal and instructions first.

In summary, the discussions so far on the GT-VITA interfaces and tutorial interfaces illustrate "what it looks like". The next chapter focuses on "how it works" with a structure walkthrough of each lesson type implemented in GT-VITA.

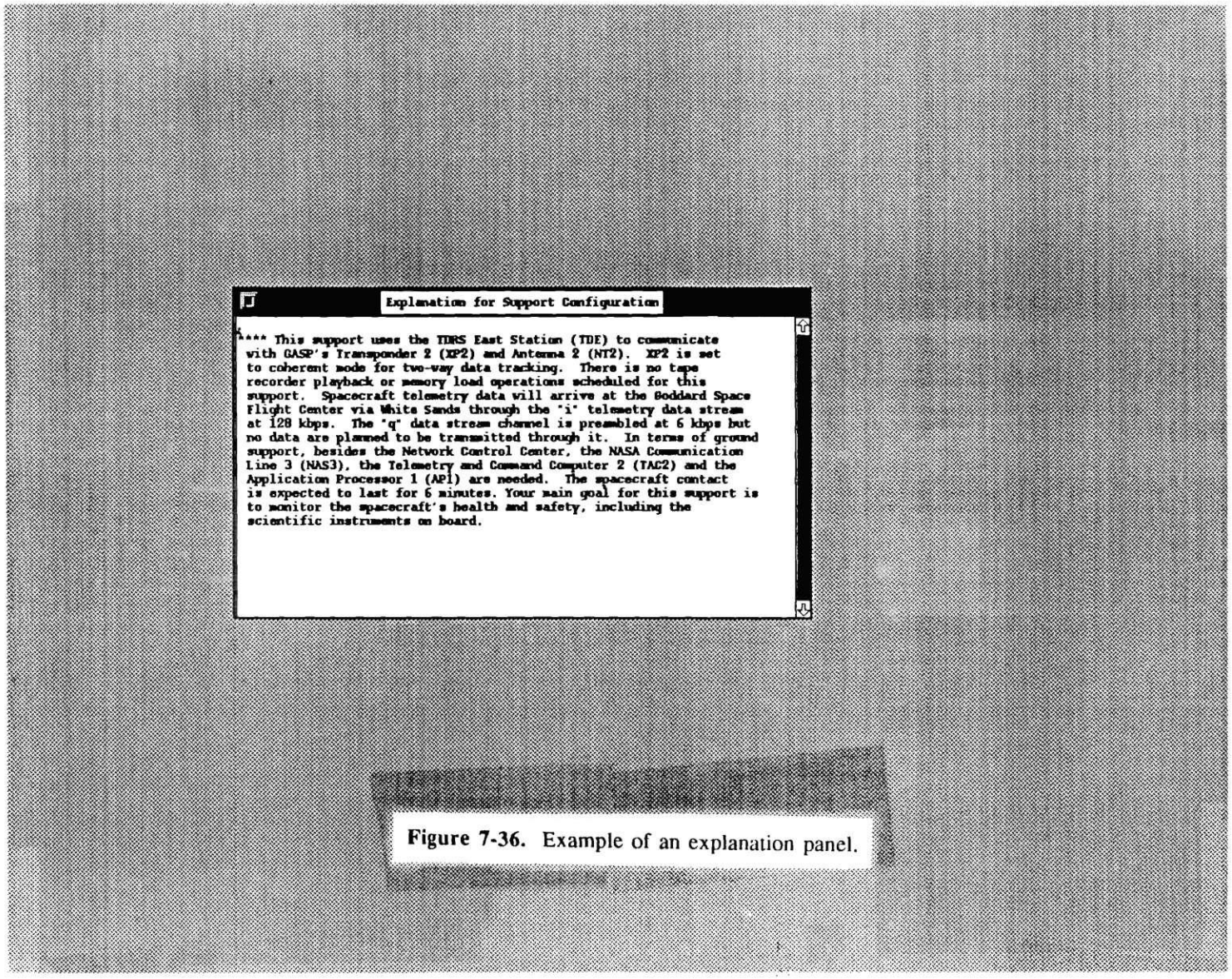


Figure 7-36. Example of an explanation panel.

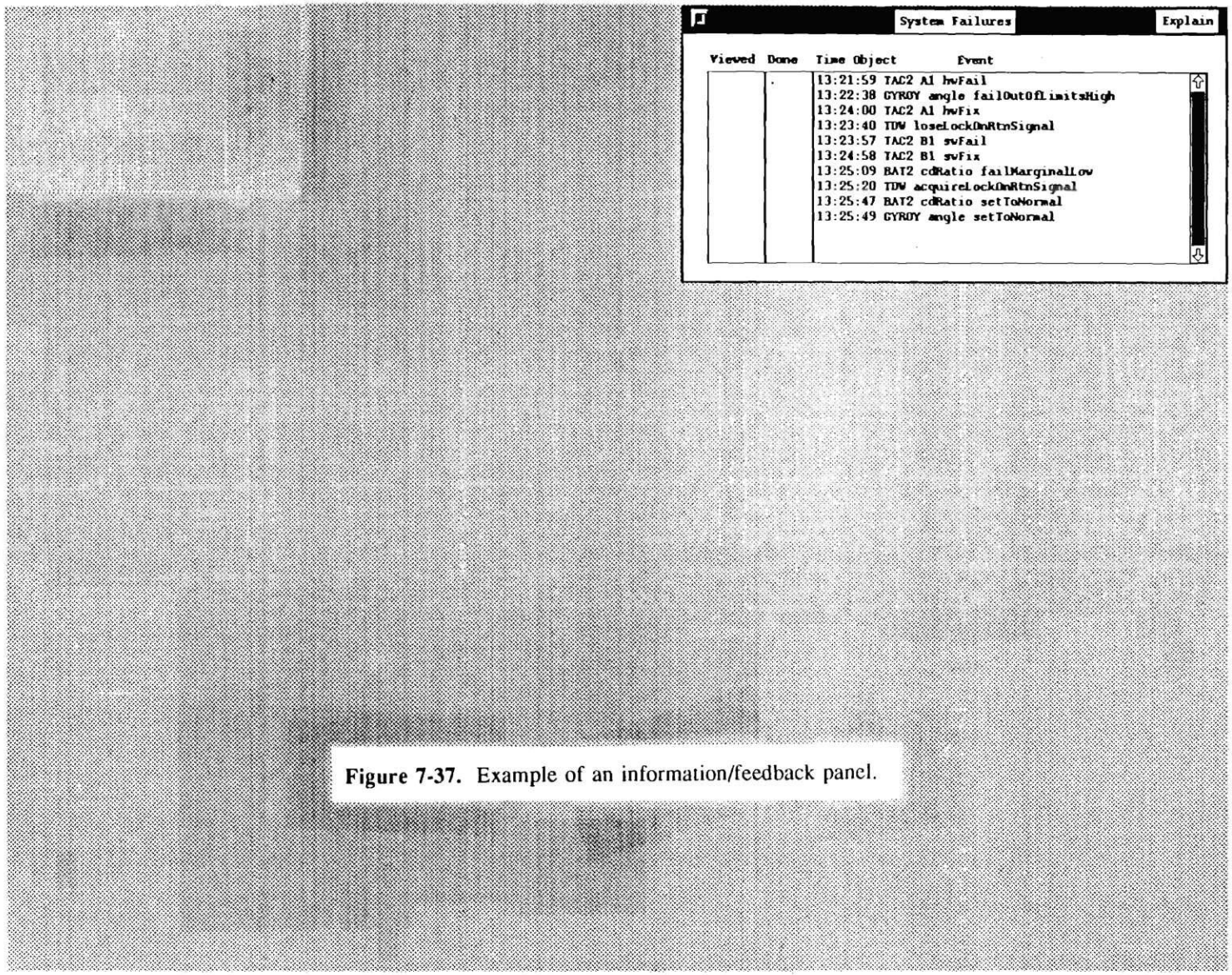
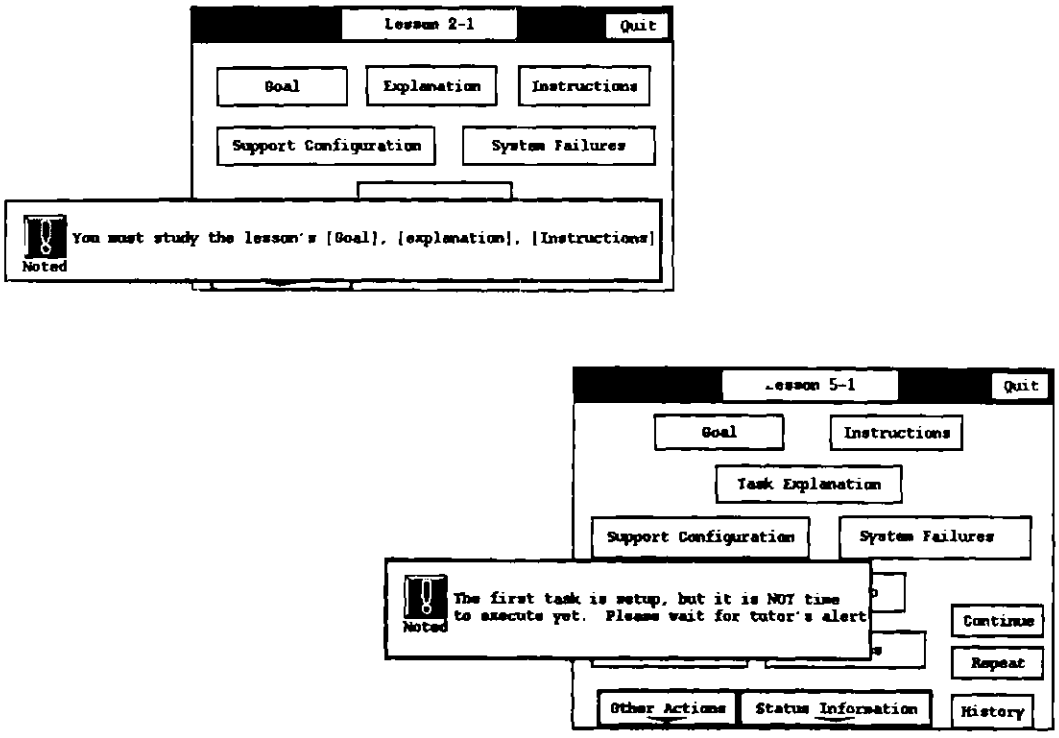


Figure 7-37. Example of an information/feedback panel.

Figure 7-38. Examples of message panels.



CHAPTER VIII

IMPLEMENTATION OF GT-VITA, THE GEORGIA TECH VISUAL INSPECTABLE TUTOR AND ASSISTANT FOR THE GEORGIA TECH PAYLOAD OPERATIONS CONTROL CENTER

PART II: STUDENT-TUTOR INTERACTIONS

This chapter continues the discussion on the implementation of GT-VITA. Below, a structured walkthrough of GT-VITA is presented by describing a sample lesson for each lesson type that is implemented in GT-VITA. For each lesson, the goal is specified in the context of GT-POCC followed by a description of typical student-tutor interactions that occurred in the lesson. For clarity and easy referencing, the figures in this section are numbered by lesson type with three levels. Figures in the first lesson type start from Figure 8-1.1, figures in the second lesson type start from 8-2.1, and so on. Also, buttons and panels that are unique to a lesson type are printed in bold.

Lesson Type 1: Learning System Components

The goal of this lesson is to introduce several high-level graphical views of the NASA system. The components in each view are also presented and described.

Example

Figure 8-1.1 shows the initial interface configuration for this lesson. The student clicks on the "Goal" and "Instructions" buttons to find out about the lesson's goal and instructions to proceed with the lesson (Figures 8-1.2 and 8-1.3). Next, the student clicks on the "**Target Panel Explanation**" to get a general description of the NASA Data/Information System -- the current target panel shown (Figure 8-1.4).

After closing the explanation panel, the student clicks on the "**Objects List**" button to get a checklist of system components that the student is supposed to learn on the target panel (Figure 8-1.5). When the student selects one of the objects in the list, an explanation panel about the object selected is shown. Moreover, the corresponding system component is highlighted in the target panel. For example, the student selects "Spacecraft", as indicated with a cross inside the checkbox. Then, a description of a spacecraft is displayed and the spacecraft object in the target panel is highlighted (Figure 8-1.6). Alternatively, the student can also get an object description by clicking on the appropriate object in the target panel. Also, there is no restriction on the order or frequency of object selection. However, the student can only attend to one object at a time: the explanation panel must be closed (after the text has been read) before a next object can be selected.

When (and only when) all the objects in the list have been attended to, the student selects the "Continue" button on the Objects List panel or on the lesson panel to proceed. In this example, another target panel is shown with a new Objects List (Figure 8-1.7). Before the student can learn about the new objects, the student is required to get a general description of the new target panel by clicking on the "Target Panel Explanation" button. Subsequently, the student proceeds as before to study every object specified in the list for the Spacecraft Subsystem panel.

The lesson ends when all target panels necessary to achieve the current lesson goal have been exhausted. When the student selects "Continue", the tutor posts a message panel to indicate the end of lesson, and another to instruct the student on how to proceed (Figures 8-1.8 and 8-1.9). The student must acknowledge both messages (one at a time) before continuing. If the student chooses to repeat the lesson, the same target panels and objects are presented again. This is the only lesson type that is not run in real time with a support scenario. For this example, the student chooses to continue and the tutor sets up for the next lesson. In between lesson setups, the tutor clears all the interfaces and displays a standby message (Figure 8-1.10).

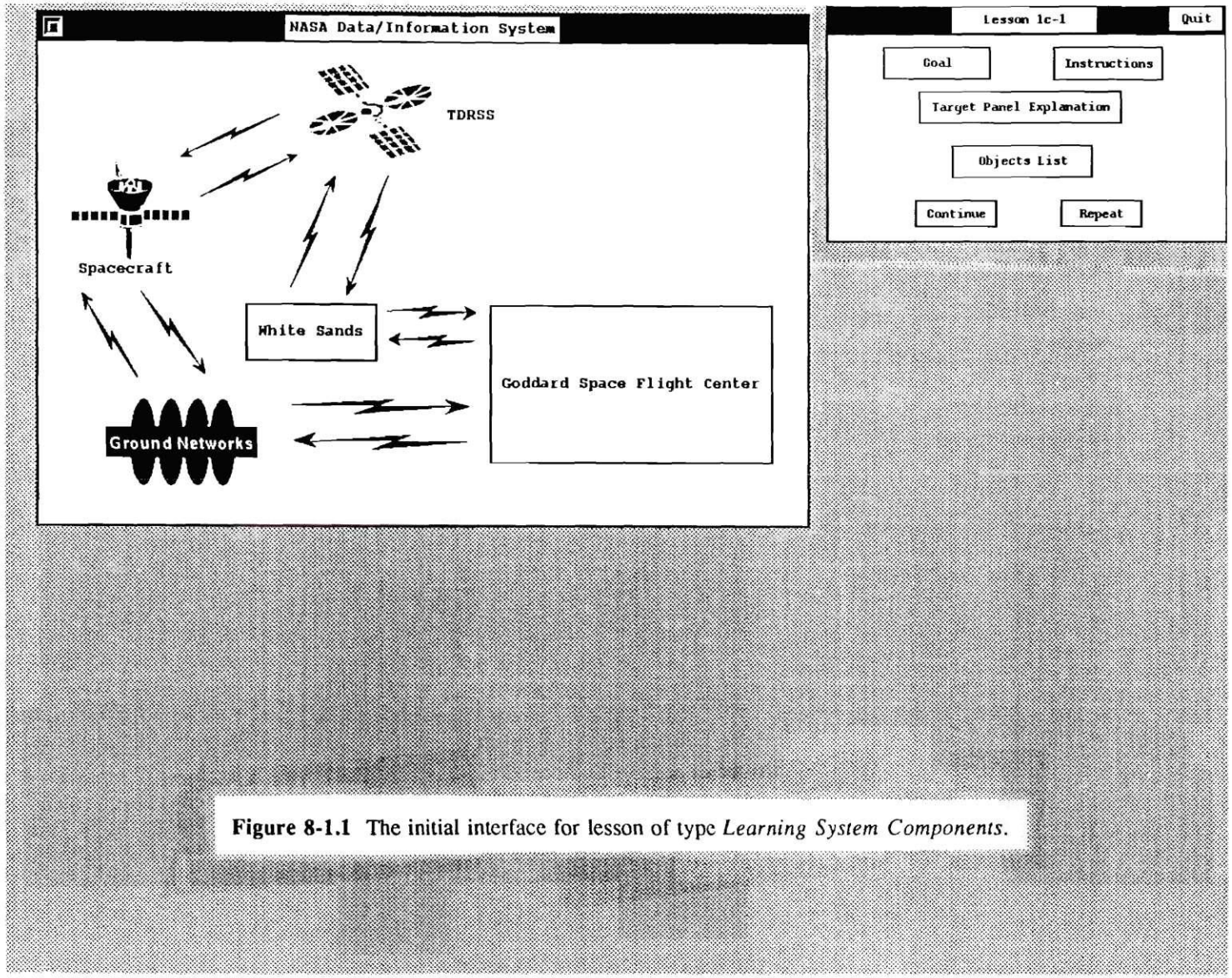


Figure 8-1.1 The initial interface for lesson of type *Learning System Components*.

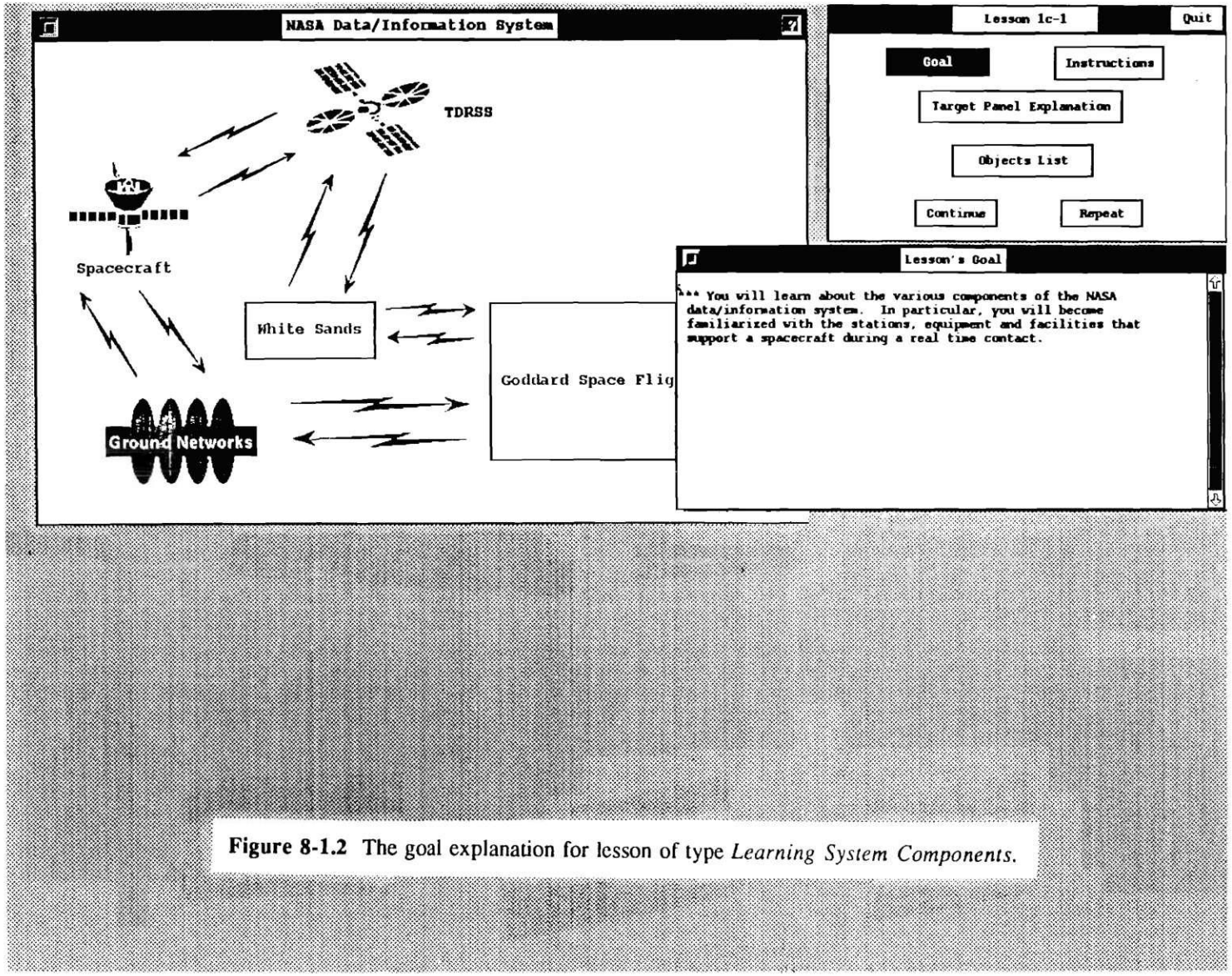


Figure 8-1.2 The goal explanation for lesson of type *Learning System Components.*

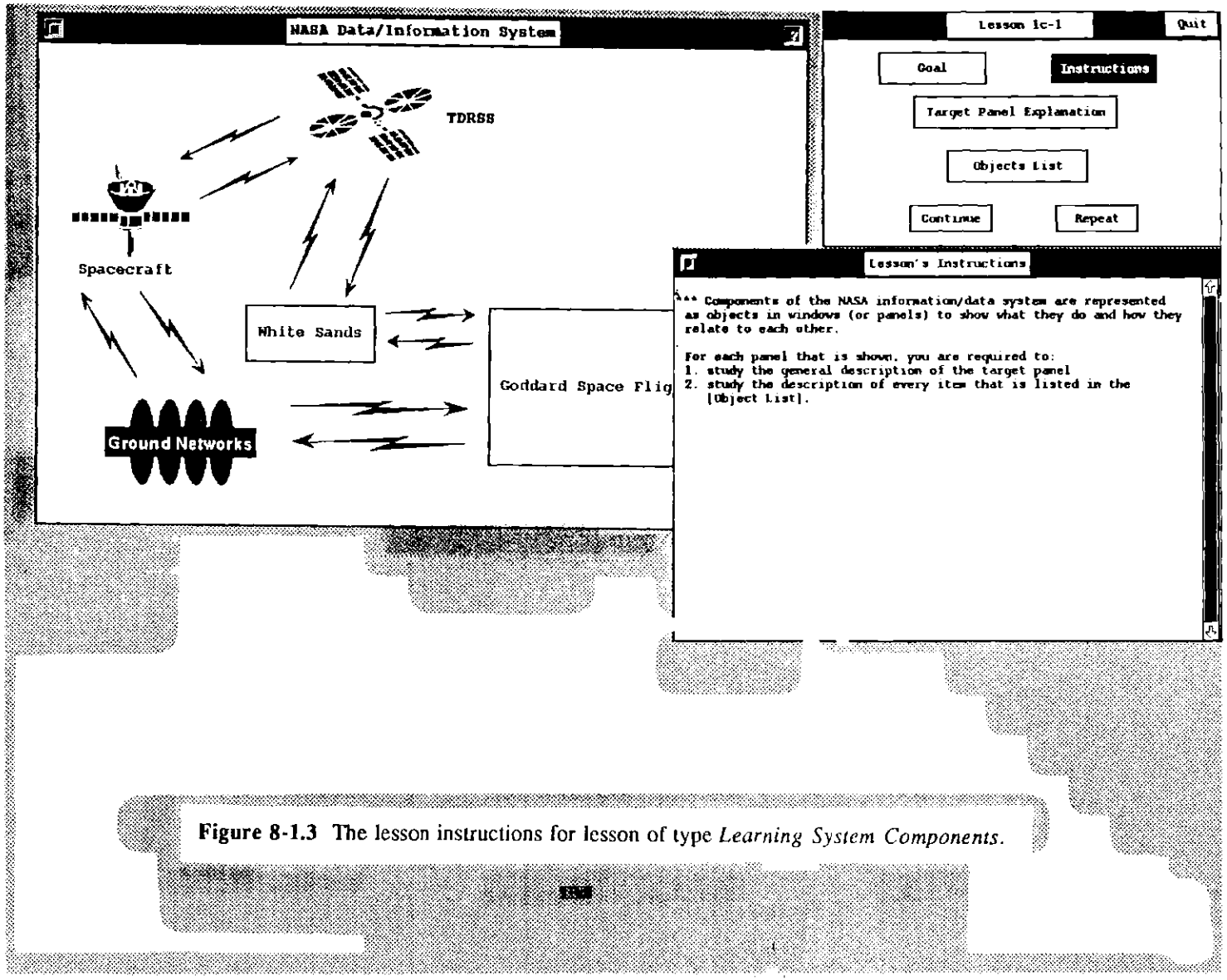


Figure 8-1.3 The lesson instructions for lesson of type *Learning System Components.*

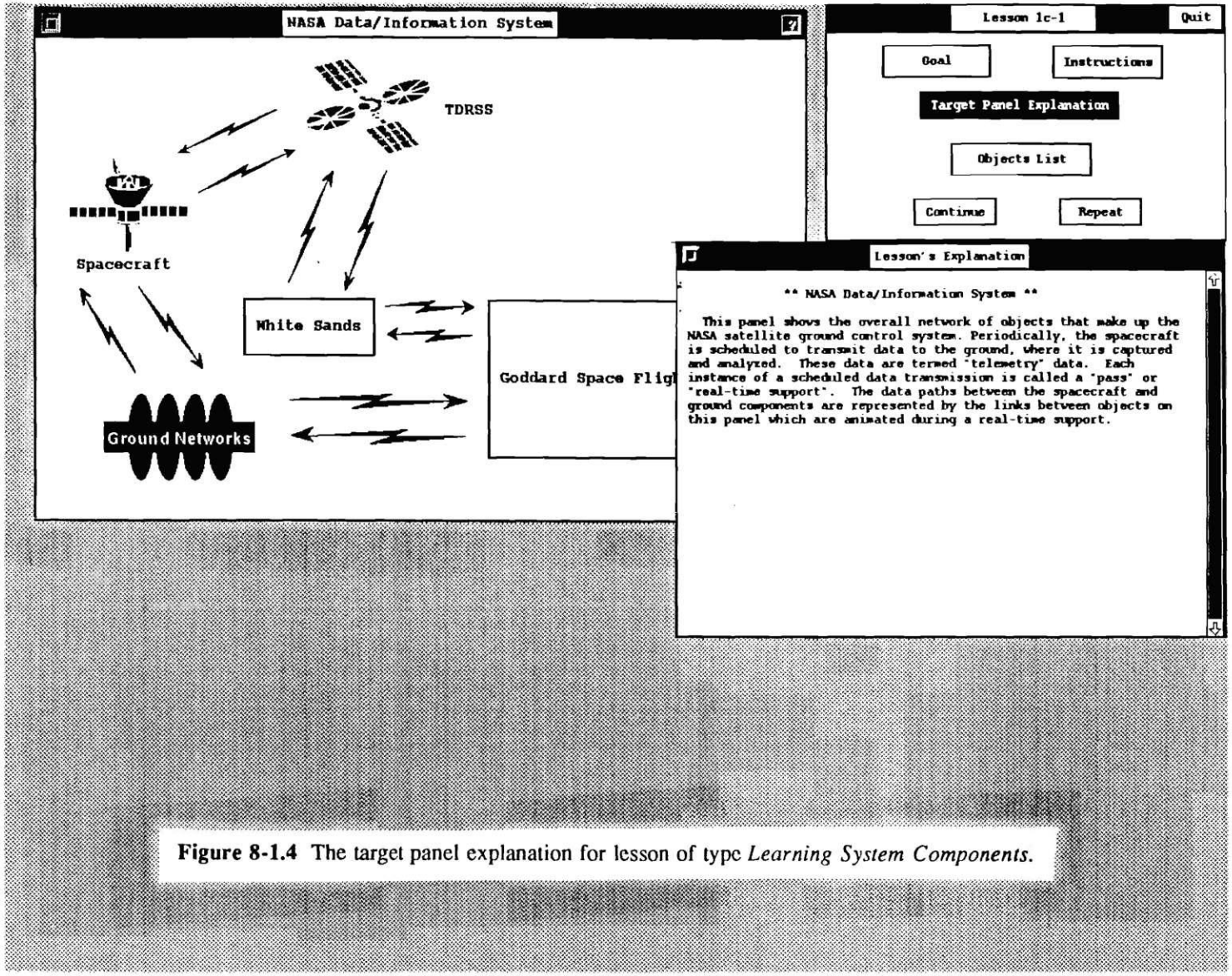


Figure 8-1.4 The target panel explanation for lesson of type *Learning System Components.*

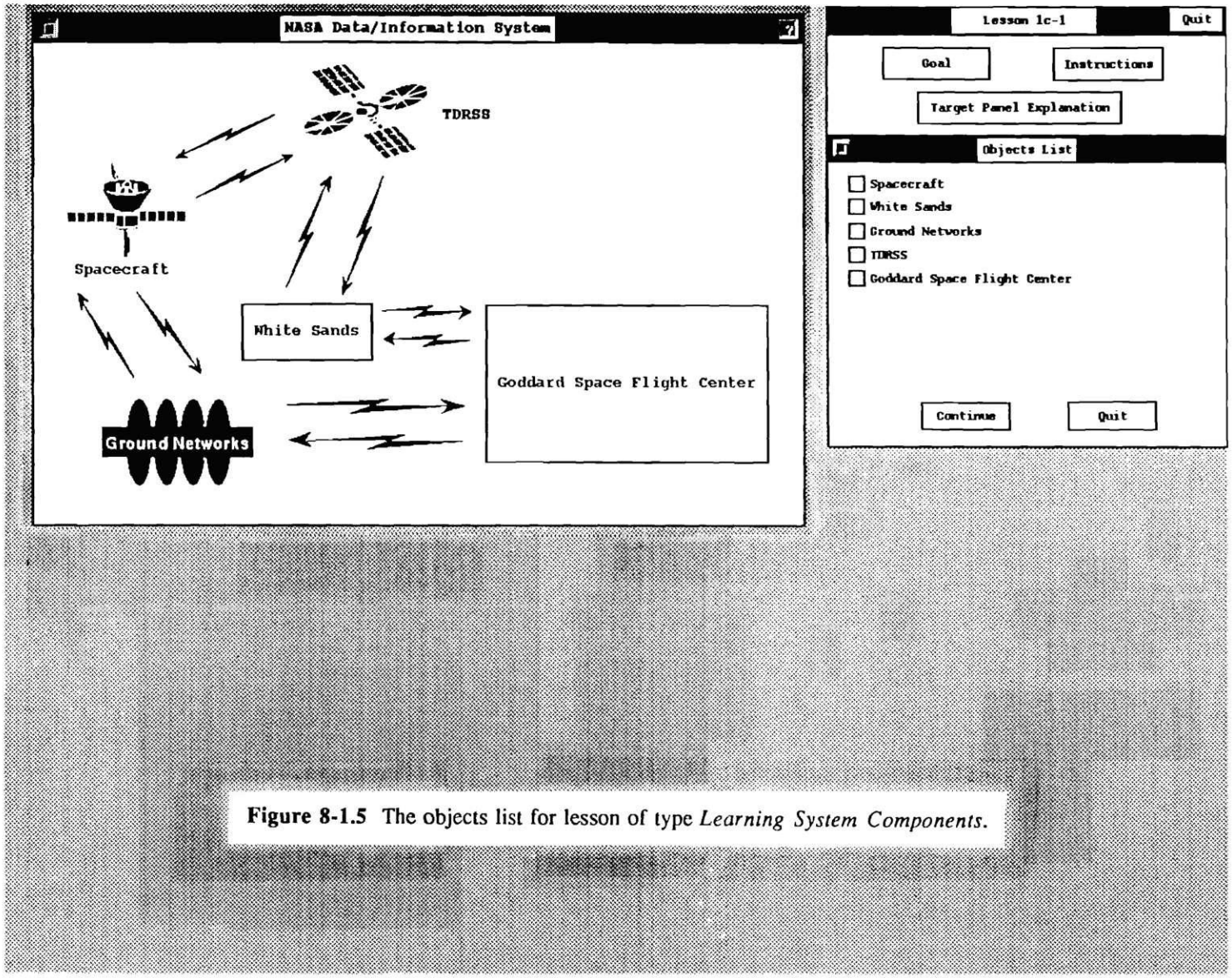


Figure 8-1.5 The objects list for lesson of type *Learning System Components.*

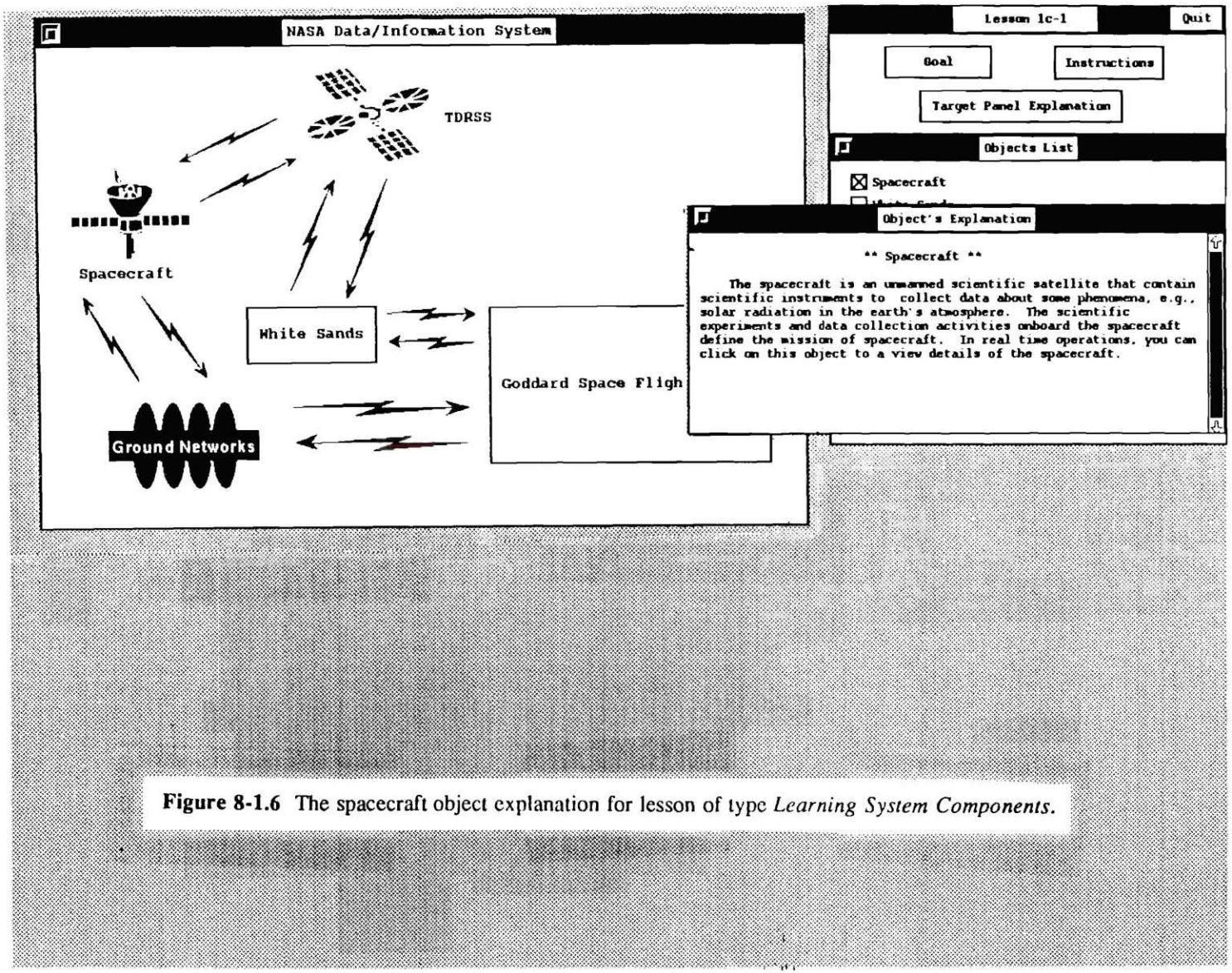


Figure 8-1.6 The spacecraft object explanation for lesson of type *Learning System Components.*

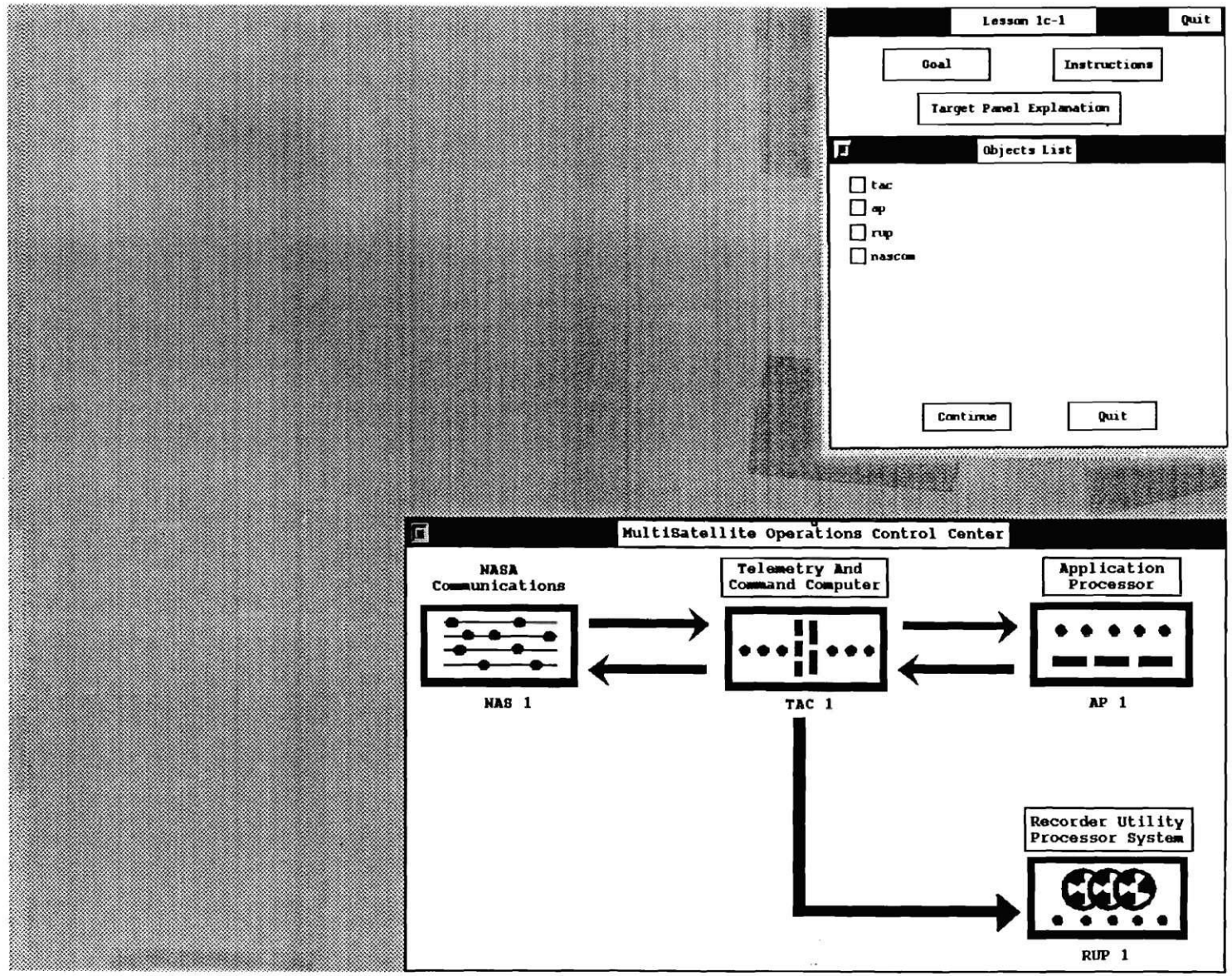


Figure 8-1.7 Another target panel and objects list for lesson of type *Learning System Components.*

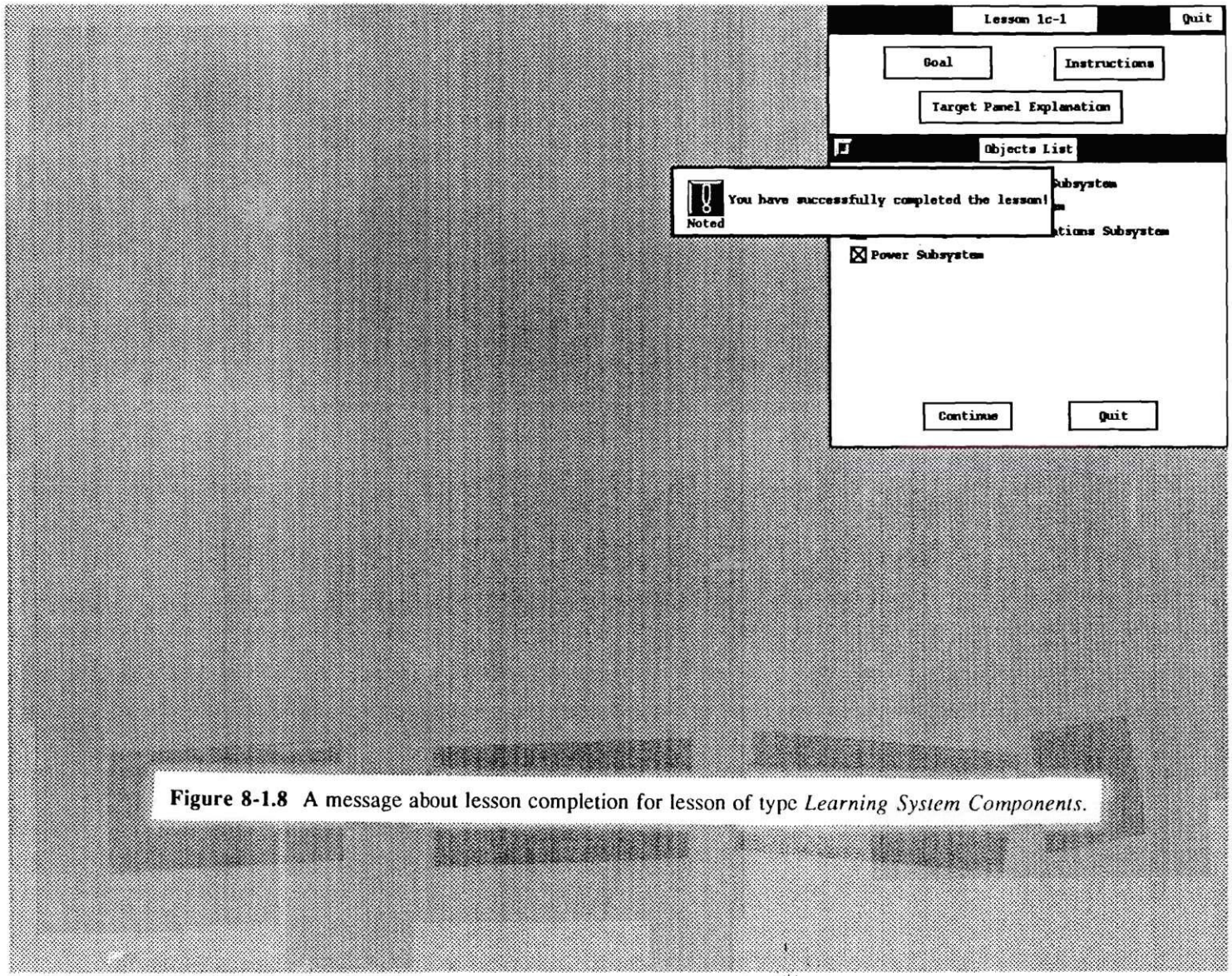


Figure 8-1.8 A message about lesson completion for lesson of type *Learning System Components.*

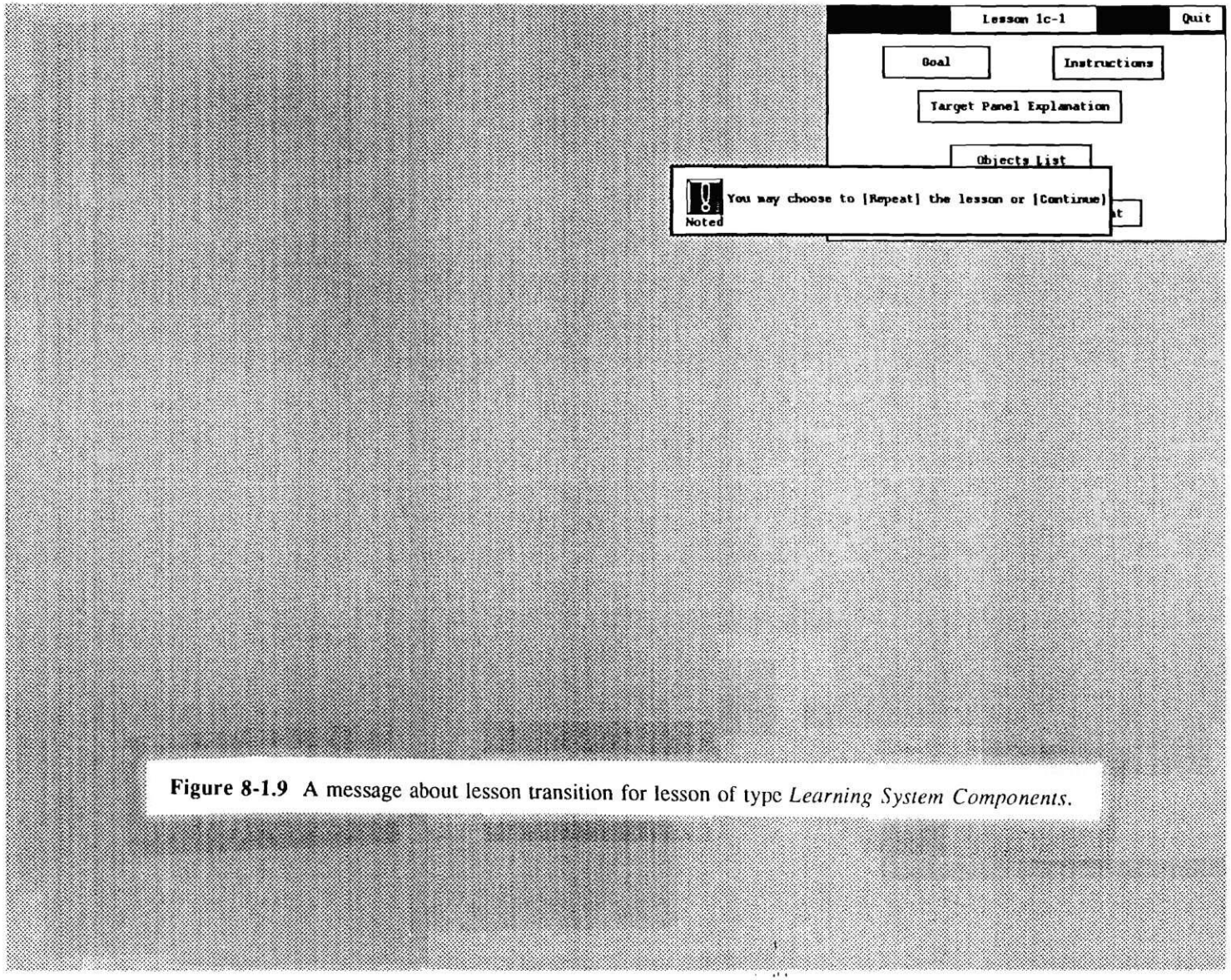


Figure 8-1.9 A message about lesson transition for lesson of type *Learning System Components*.

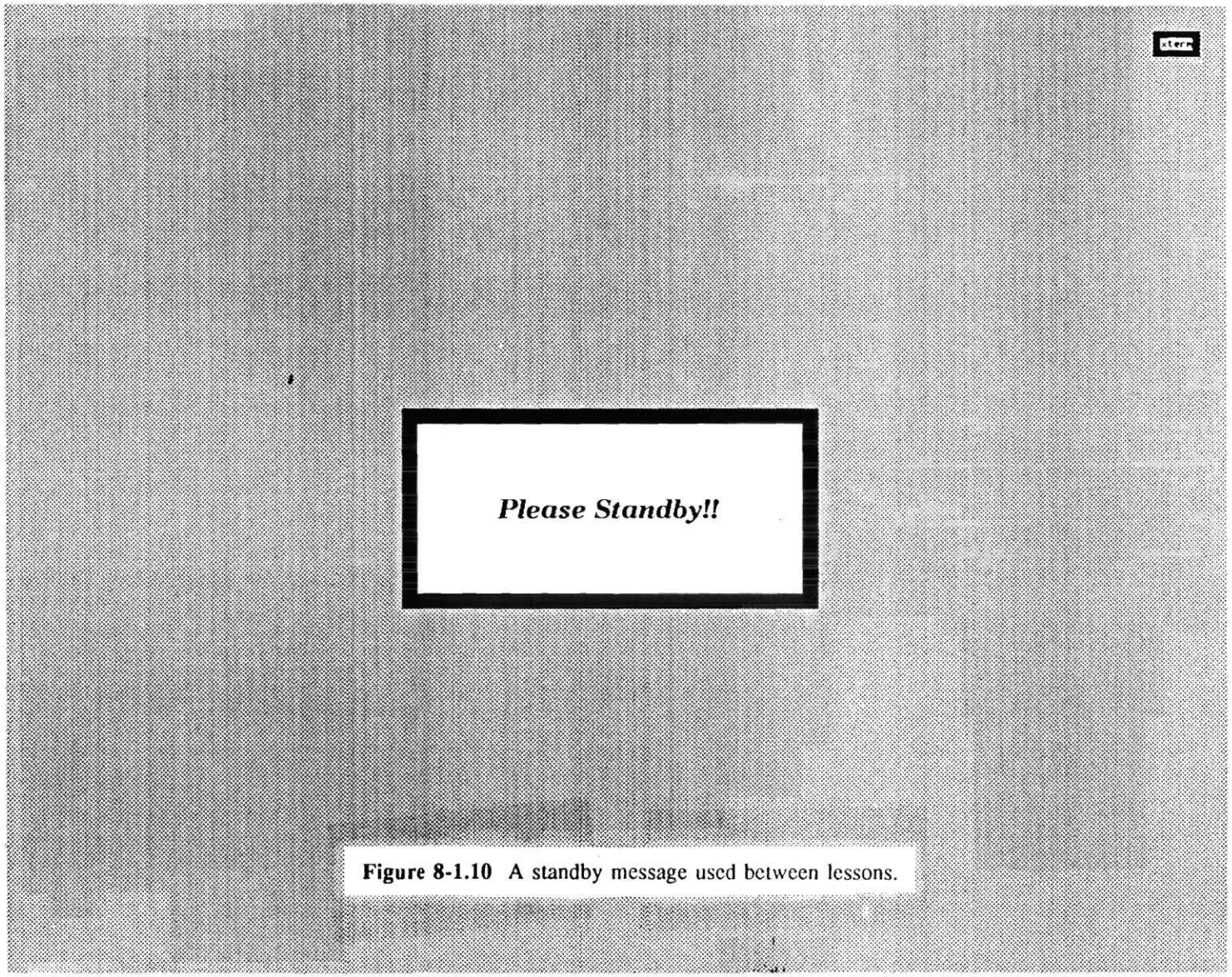


Figure 8-1.10 A standby message used between lessons.

Lesson Type 2: Learning System Behavior

The goal of this lesson is to introduce new concepts that influence the behavior of the NASA system in real time. They are: support configuration, data flow, and failures. The tutor not only describes these concepts but also demonstrates them by animating the data flow from the spacecraft to each ground component, and the effects of failure events on system behavior.

Example

After viewing the lesson's goal, the student clicks on the "Explanation" button which displays a description of the new concepts on support configuration, data flow and failures (Figure 8-2.1). When the student clicks on the "Support Configuration" button (after attending to the first row), a panel by the same name displays specific information about the support that define the scenario for the current lesson. To understand the information presented in the support configuration, the student clicks on the "Explain" button to read a textual description about the support (Figure 8-2.2). Next, the student selects the "**System Failures**" button on the lesson panel to display a panel of the same name (Figure 8-2.3). This new panel shows a list of failure events (and their "fixes") that the tutor plans to demonstrate to the student at the specified times. For now, to get a general idea of what the set of failures are about, the student clicks on the "Explain" button for more information. To learn about system failures and data flow, the lesson scenario must be run in real time. Thus, the student selects the "Start Scenario" button on the lesson panel, selects the "Show NASA Network" button and finally starts the scenario (Figure 8-2.4).

Once the scenario starts, the student can explore the system by inspecting different objects at different levels of details as discussed previously. However, in order to promote the correct interpretation of the graphical elements for representing system behavior, the tutor provides a set of explanations to describe the data flow representations on various panels. These explanations are available through the pulldown menu labeled "**Explain Flow**" on the lesson panel. Figure 8-2.5 shows the selection of panel names for the pulldown menu. Figure 8-2.6 shows the data flow explanation for the NASA Data/Information System.

Figure 8-2.7 shows the data flow explanation for the Command & Data Handling Subsystem. The student is required to view all data flow explanations that the tutor provides.

Similarly, to promote understanding of system failures, the tutor provides explanations for each event listed in the System Failures panel in terms of the problem definition, the effects and symptoms attributed to the failure, the appropriate operator action to rectify the problem (when control actions are permitted), and the panel(s) on which the failure is graphically represented. For example, before the occurrence of a scheduled failure, the student clicks on the failure item which is then highlighted, and then clicks on the "Explain" button to get an explanation panel for the selected failure. Subsequently, the student watches the state change in the corresponding system panels as the failure occurs. The tutor keeps track of the number of times each failure item has been attended to in the column labeled "Viewed". The tutor also marks a failure event as it occurs with three asterisks in the column labeled "Done" (Figure 8-2.8). In keeping with the lesson's goal to learn system failures, the student is required to view each failure item at least once and all failure items must occur before a lesson is considered complete. The student has the option to pause and restart the support scenario anytime during the lesson. This option is especially useful for this lesson type because the student may miss system events that occurred while busily reading explanations on flows or failures. The pause/restart option enables the student to pace the reading with real-time system events.

As part of exploring the system, the student clicks on the tape recorder object on the Command & Data Handling Subsystem which displays the control panel for the tape recorder. The student next attempts to change the rate value from high to low. The tutor reacts to the student's action with the message indicating that control actions are not permitted at the moment. Figure 8-2.9 shows this situation.

When the lesson scenario ends, the tutor alerts the student with a message as shown in Figure 8-2.10. The student may decide to repeat the lesson to study more about system behavior with another support scenario and set of failure events. Or, as in the present example, the student decides to move on. After the student selects the "Continue" button on the lesson panel, the tutor diagnoses the student's performance to assess if all lesson requirements have been met. If the student has not attended to the descriptions for the support configuration, all failure events, or all data flow paths, the tutor alerts the

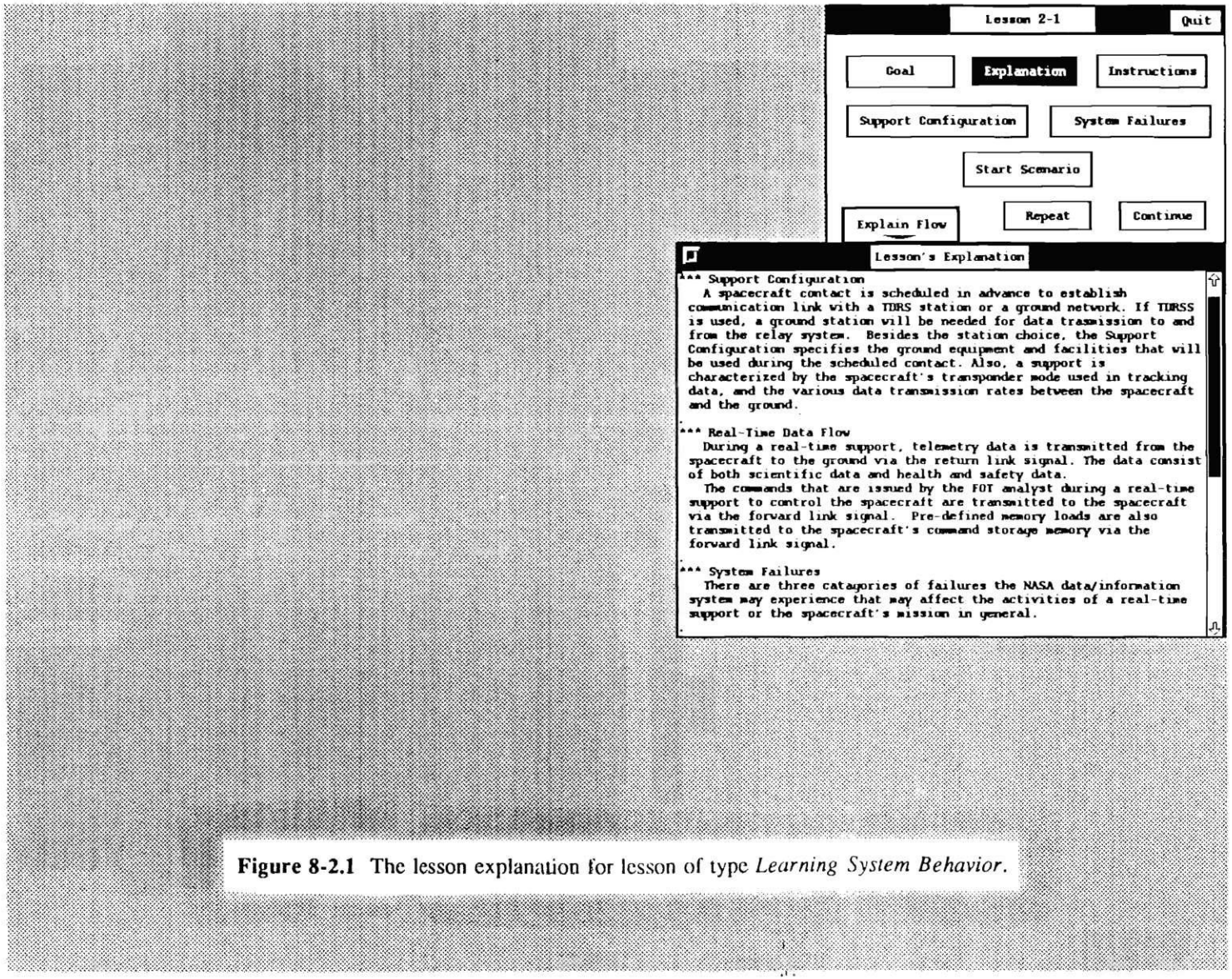


Figure 8-2.1 The lesson explanation for lesson of type *Learning System Behavior.*

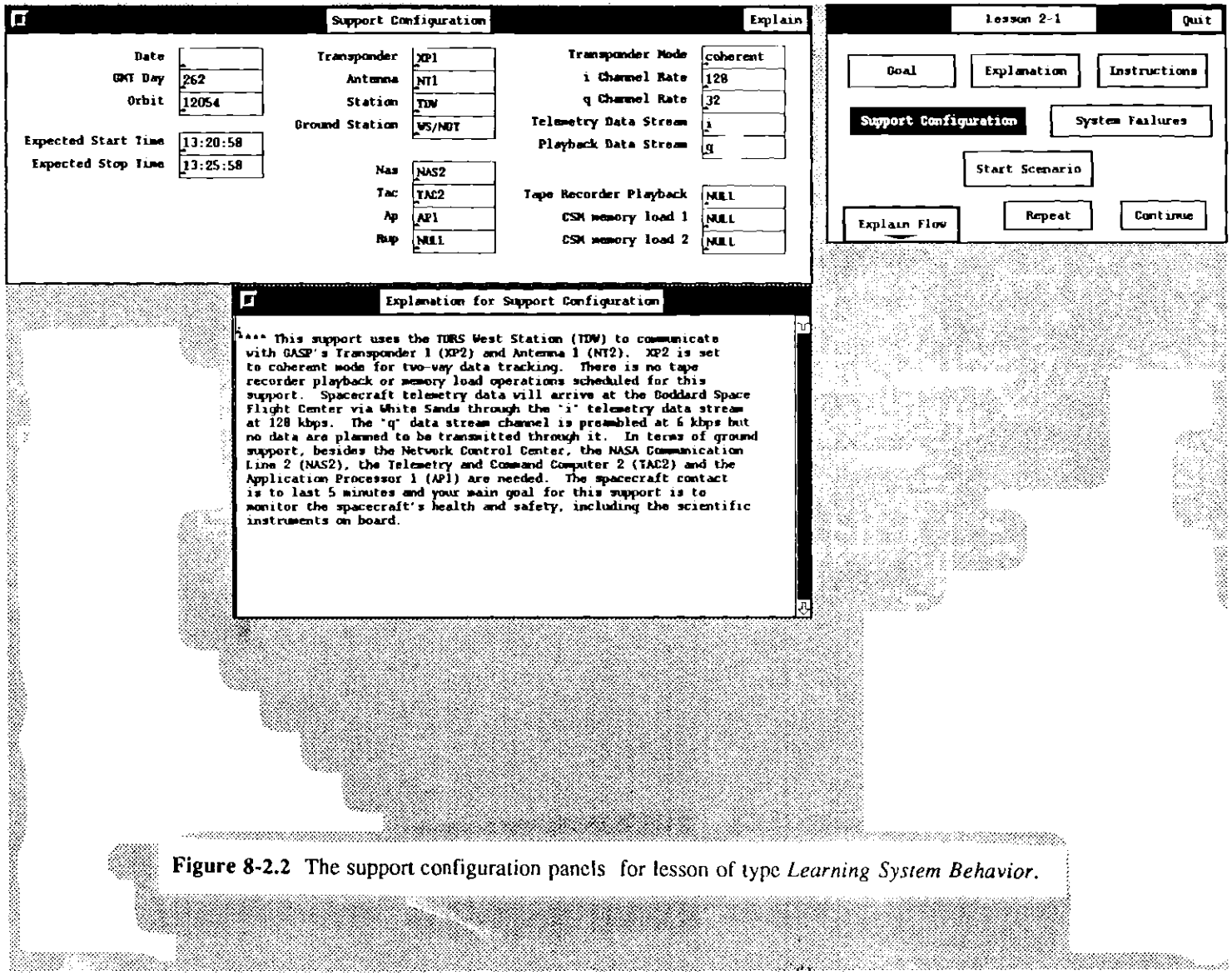


Figure 8-2.2 The support configuration panels for lesson of type *Learning System Behavior.*

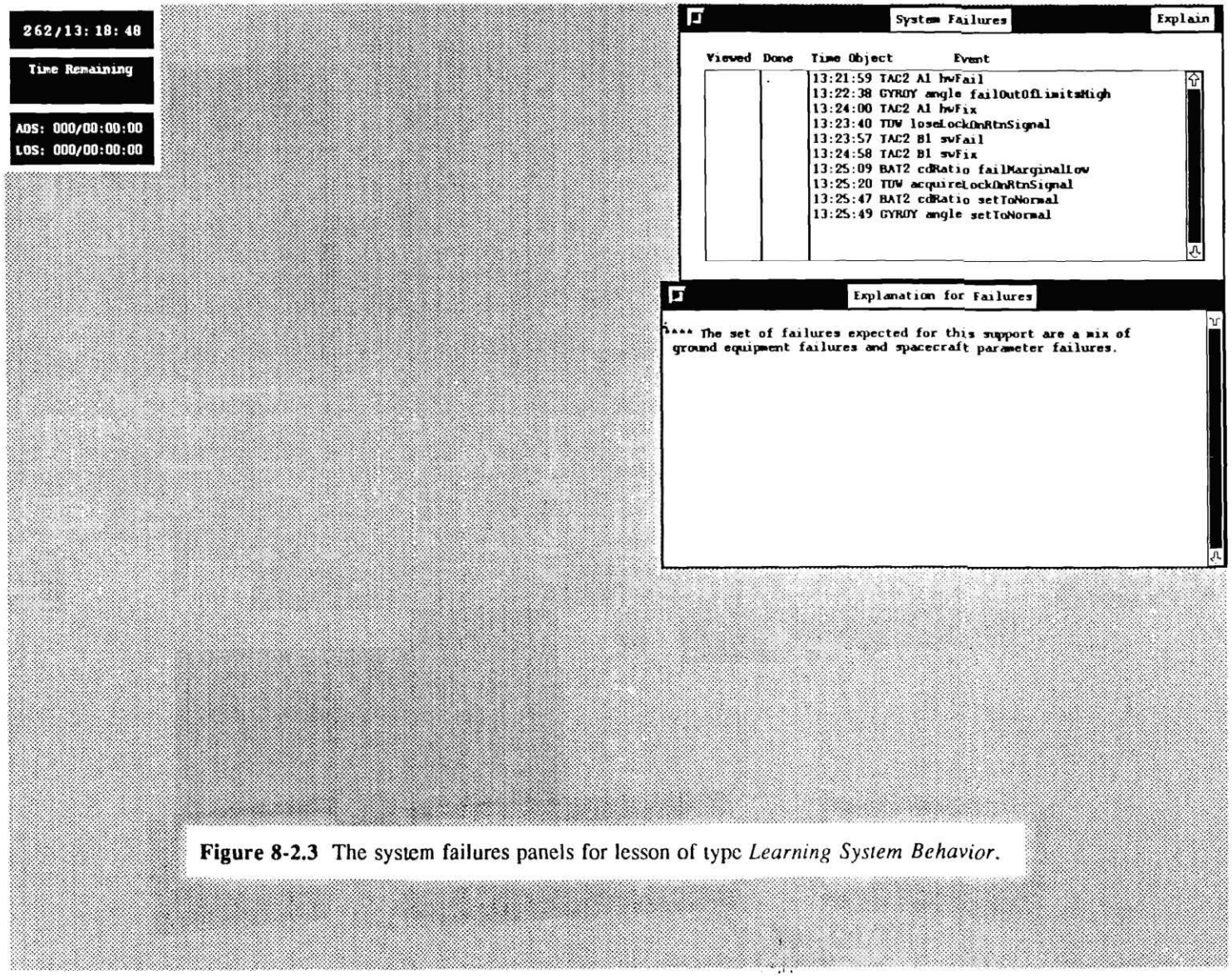


Figure 8-2.3 The system failures panels for lesson of type *Learning System Behavior*.

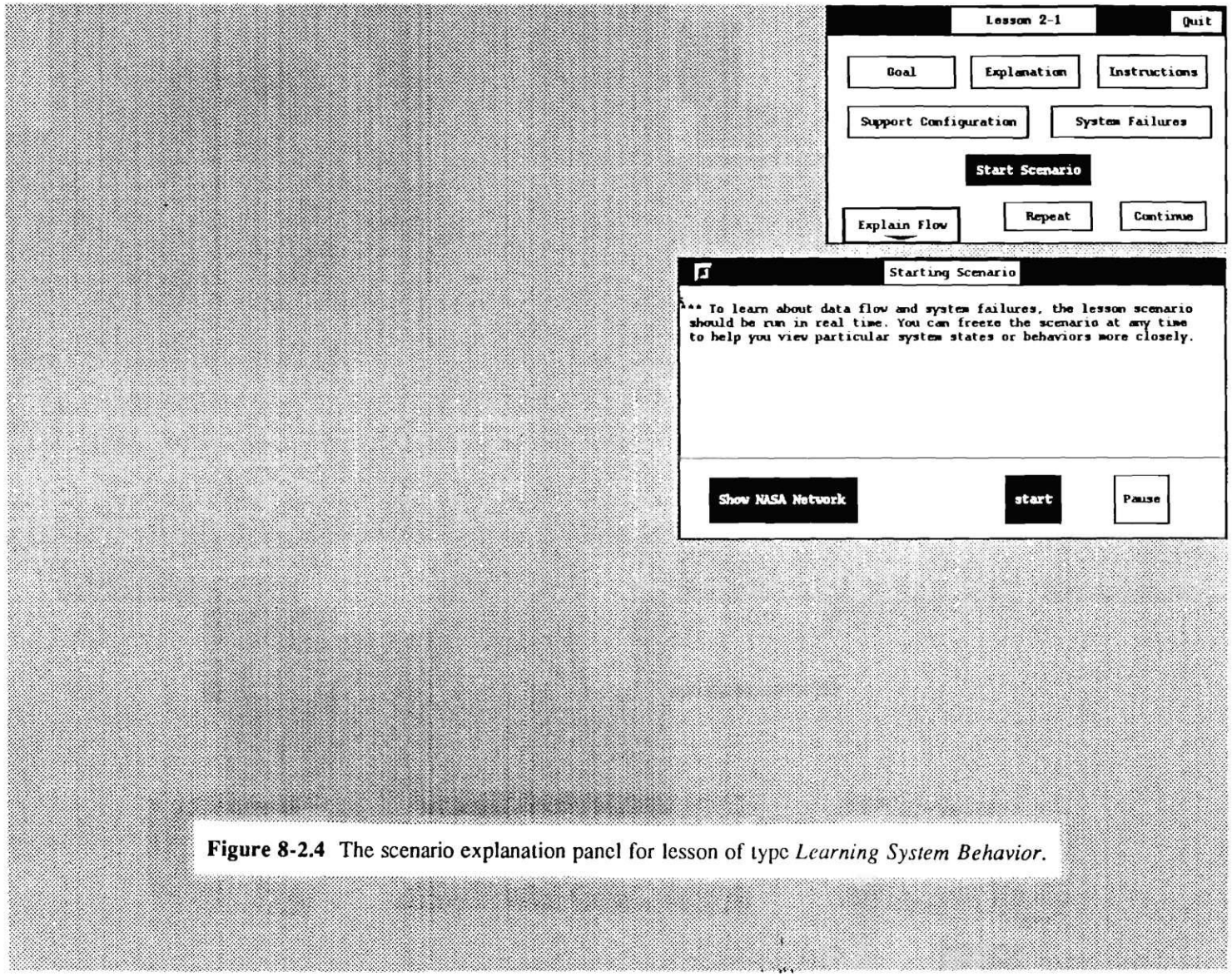


Figure 8-2.4 The scenario explanation panel for lesson of type *Learning System Behavior.*

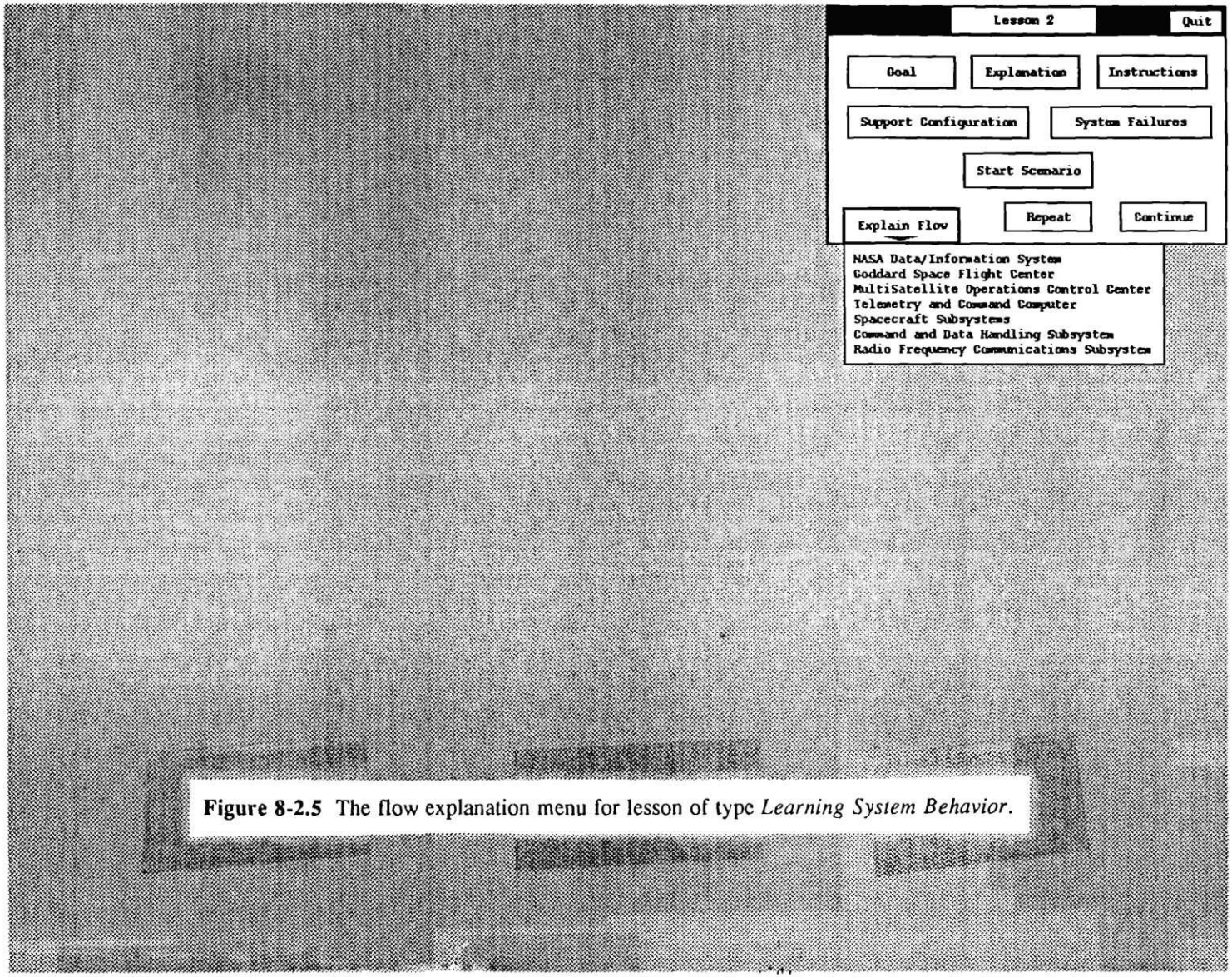


Figure 8-2.5 The flow explanation menu for lesson of type *Learning System Behavior*.

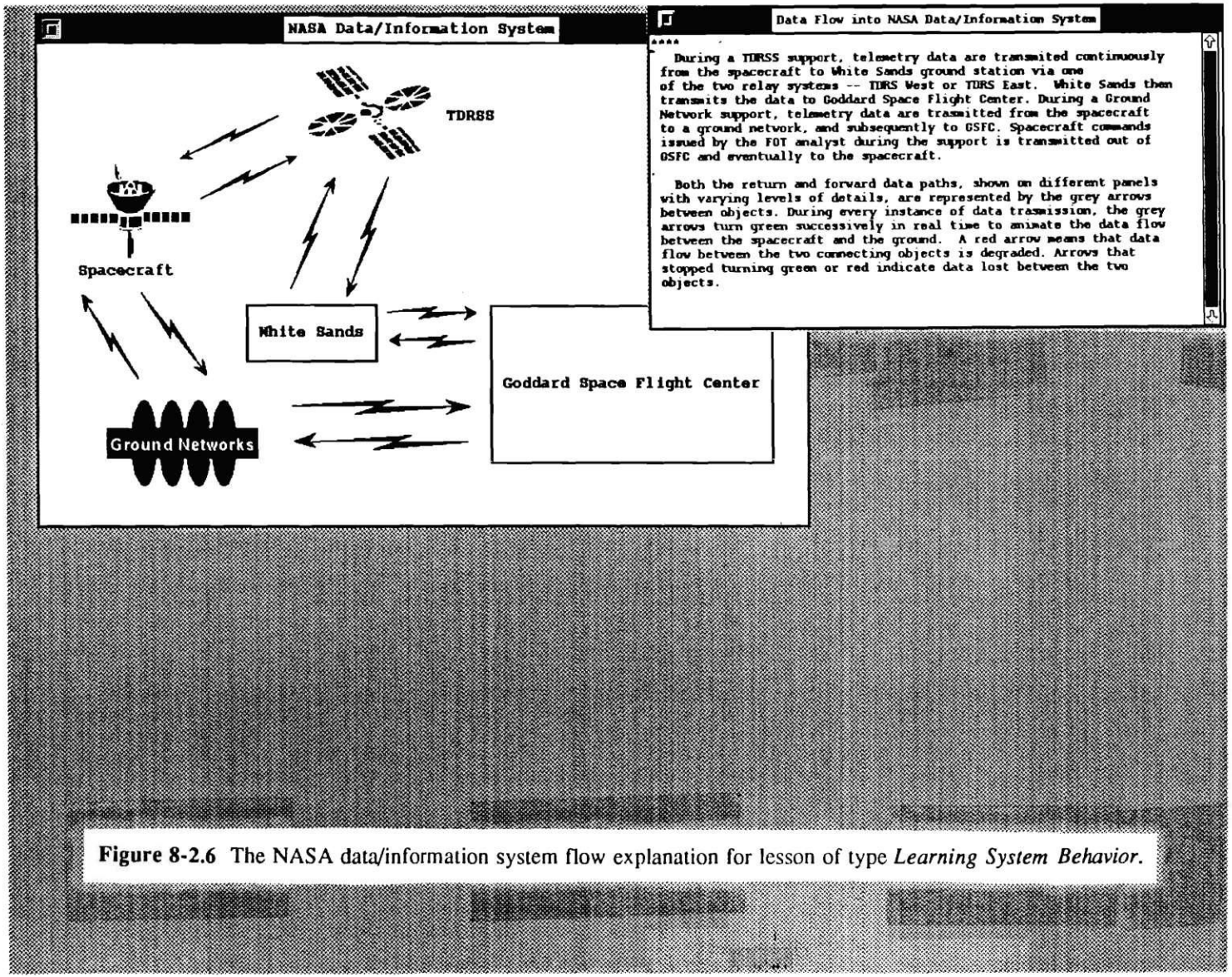


Figure 8-2.6 The NASA data/information system flow explanation for lesson of type *Learning System Behavior.*

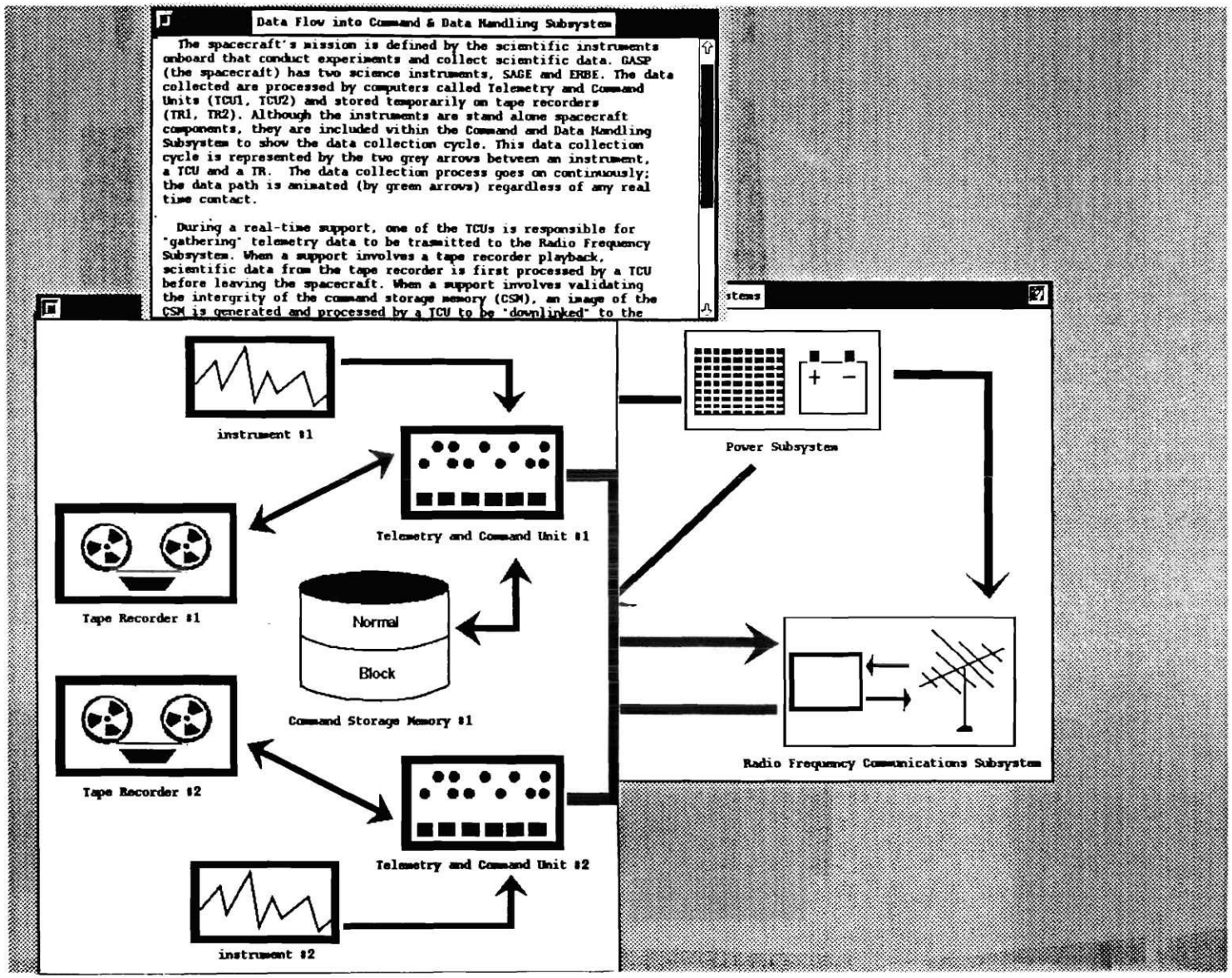


Figure 8-2.7 The command and data handling subsystem flow explanation for lesson of type *Learning System Behavior*.

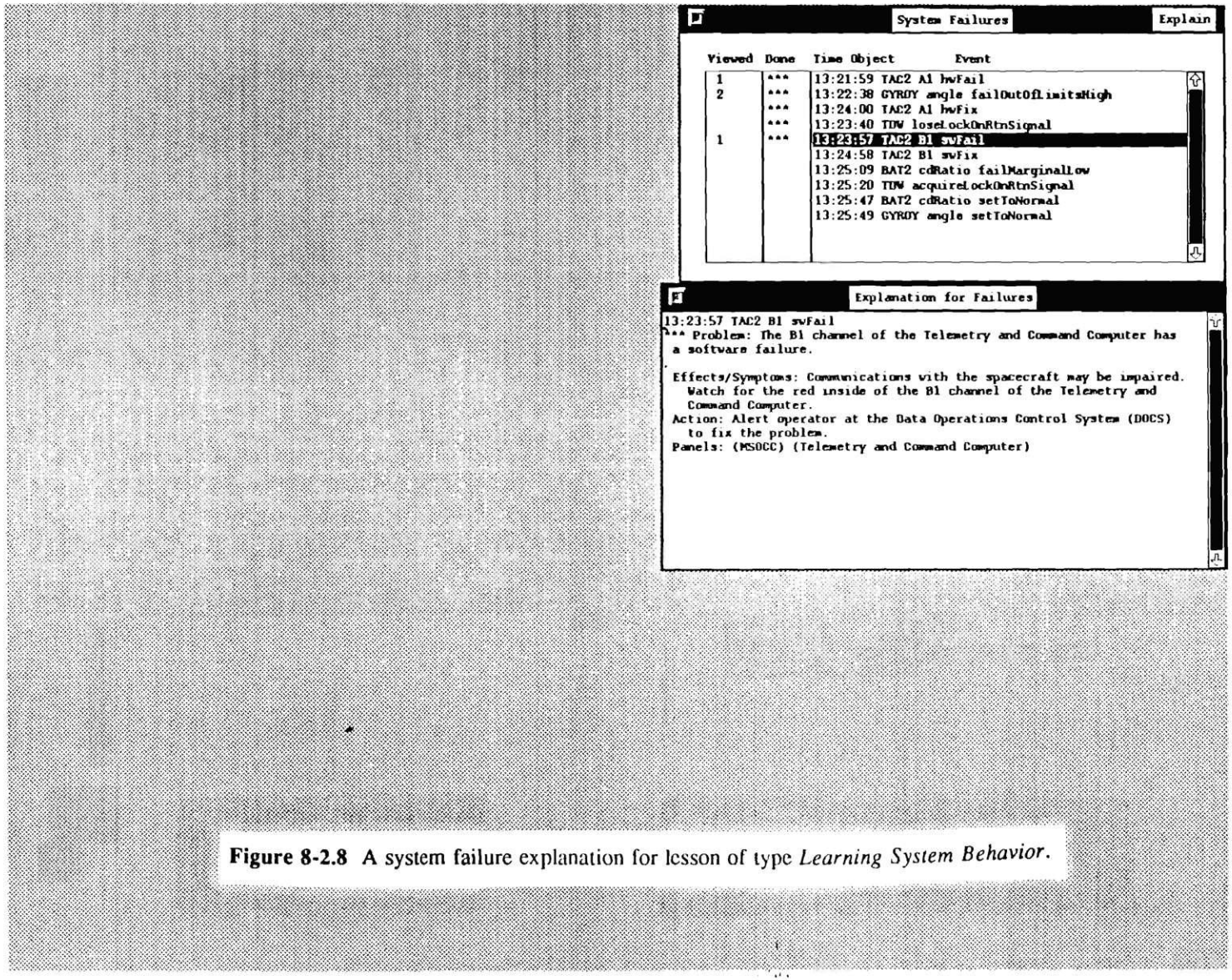


Figure 8-2.8 A system failure explanation for lesson of type *Learning System Behavior.*

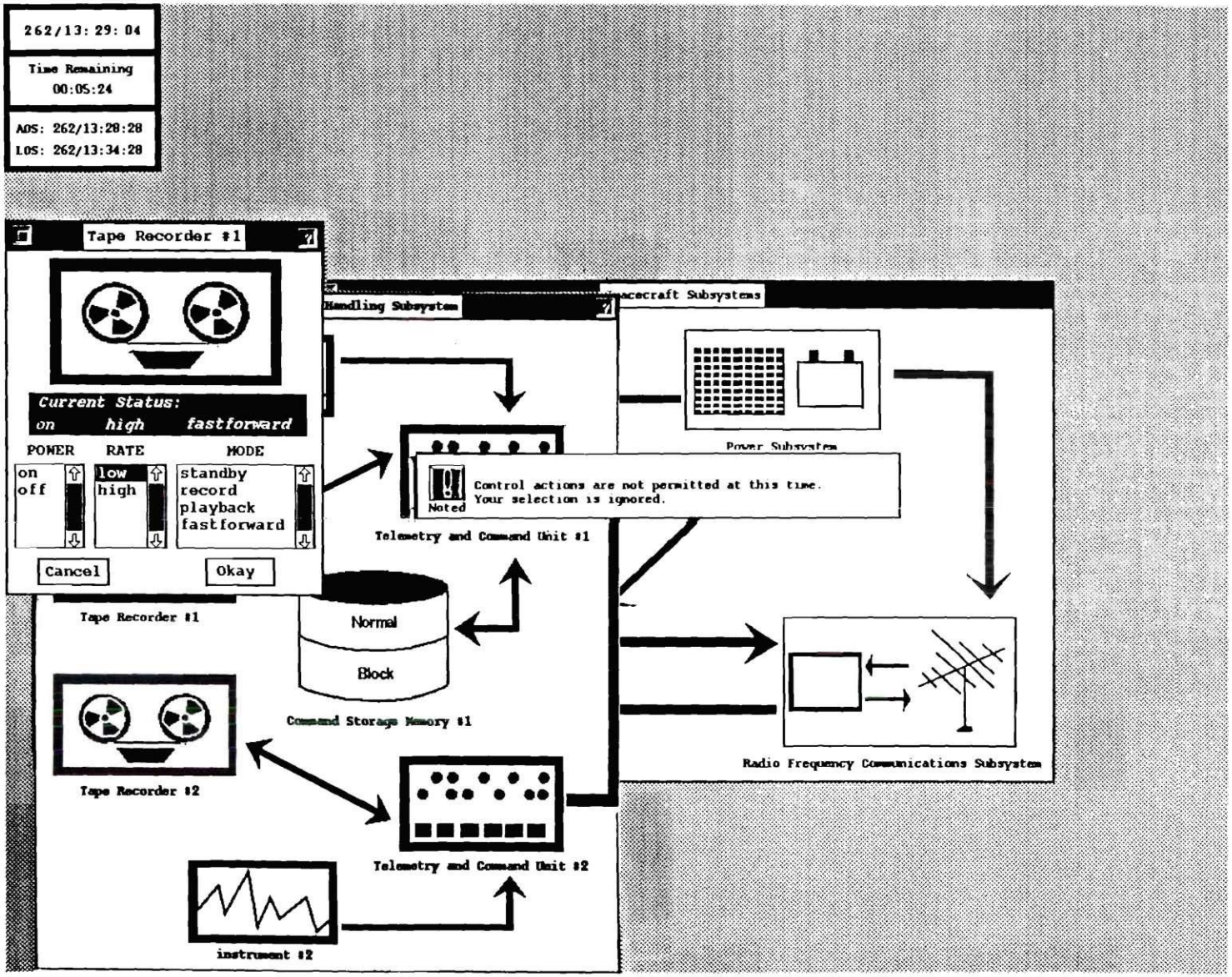


Figure 8-2.9 A message indicating invalid action for lesson of type *Learning System Behavior*.

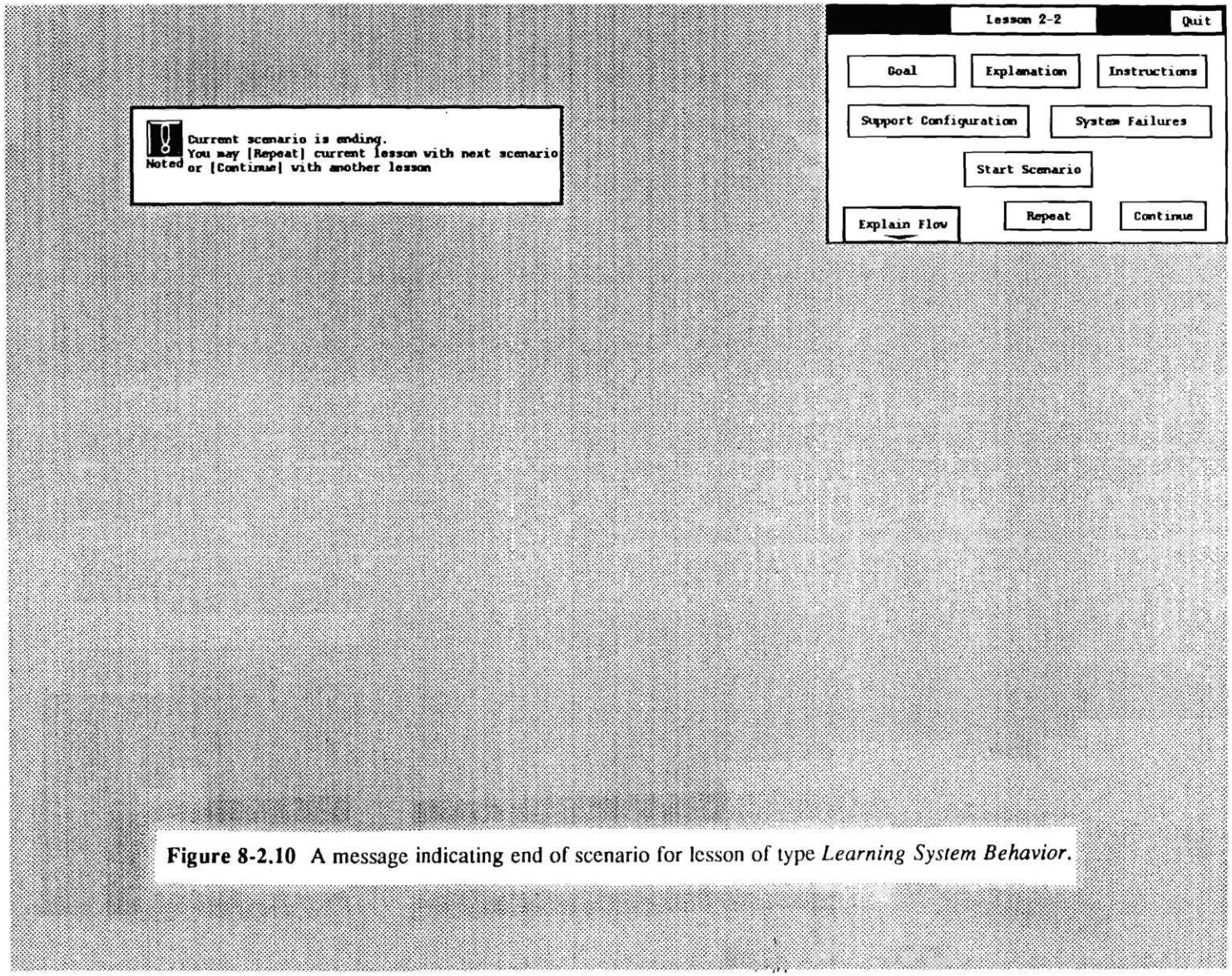


Figure 8-2.10 A message indicating end of scenario for lesson of type *Learning System Behavior*.

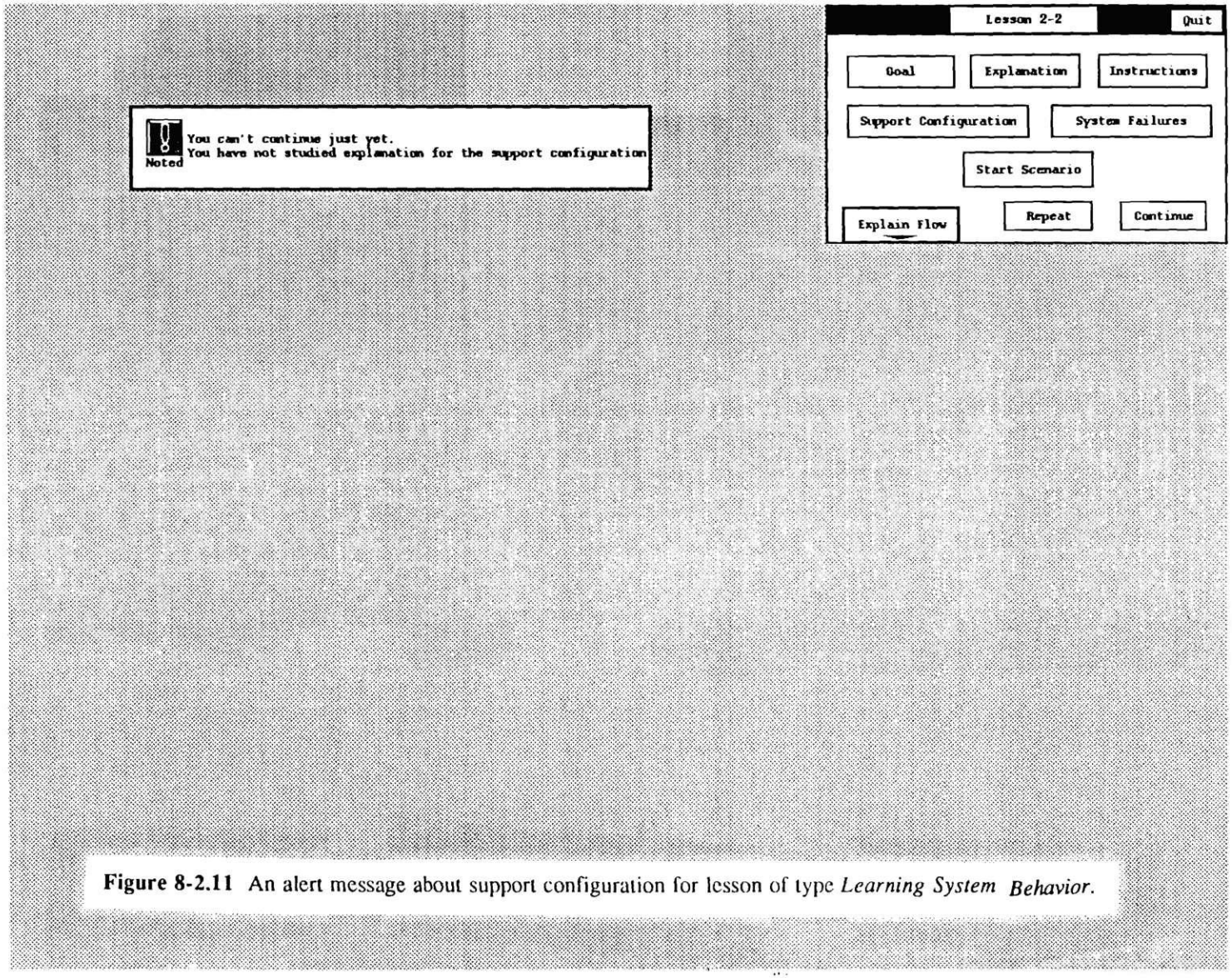


Figure 8-2.11 An alert message about support configuration for lesson of type *Learning System Behavior.*

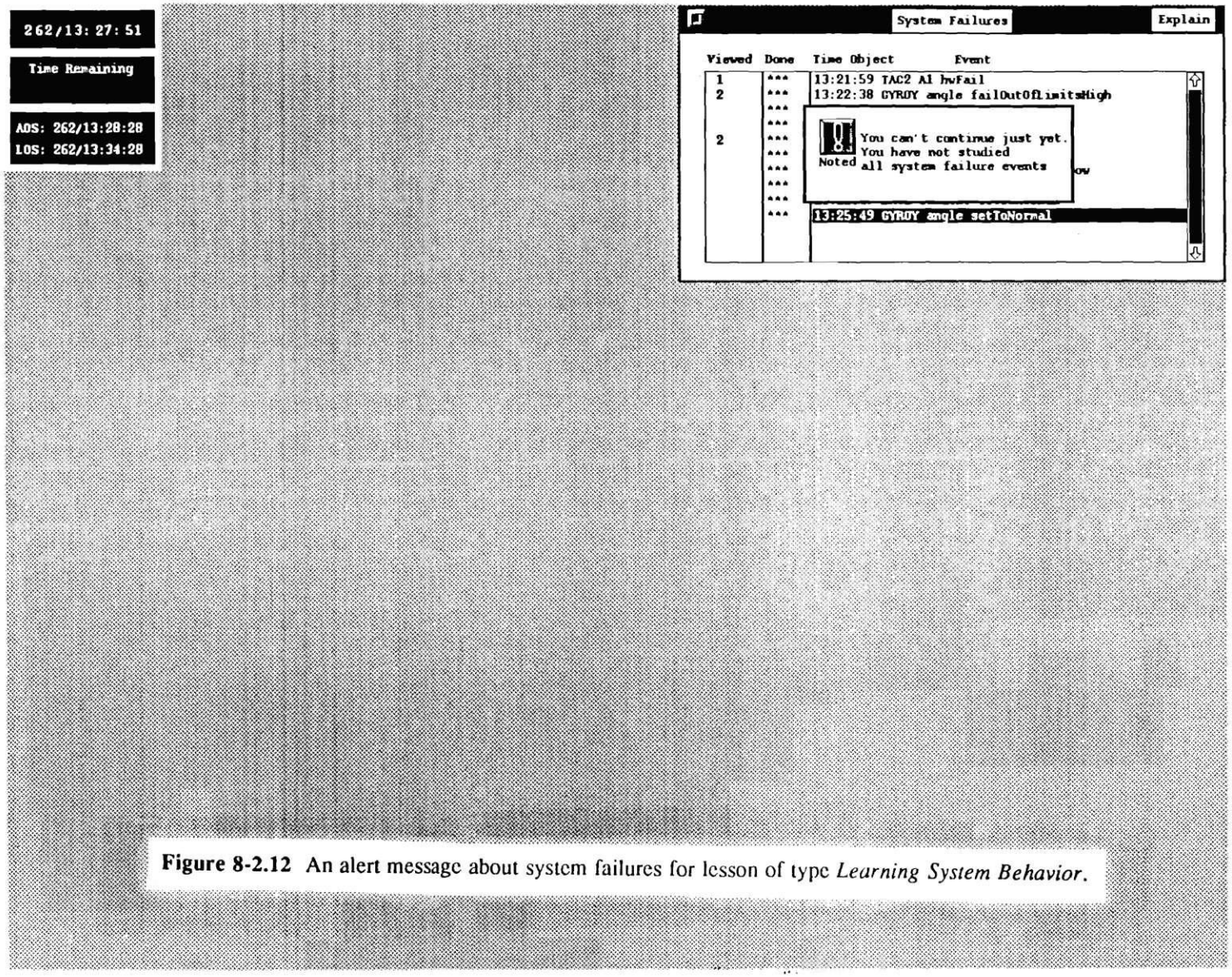


Figure 8-2.12 An alert message about system failures for lesson of type *Learning System Behavior.*

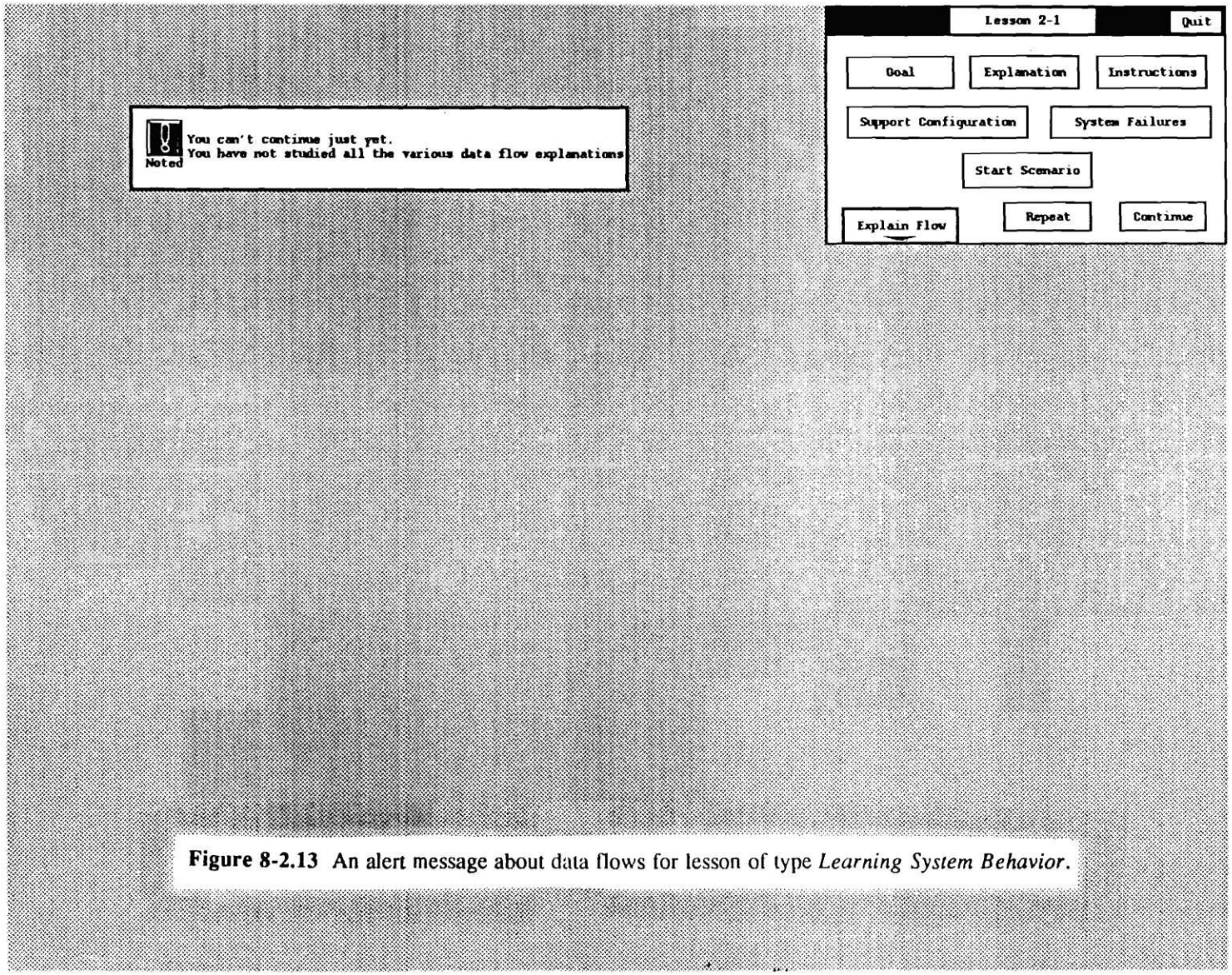


Figure 8-2.13 An alert message about data flows for lesson of type *Learning System Behavior*.

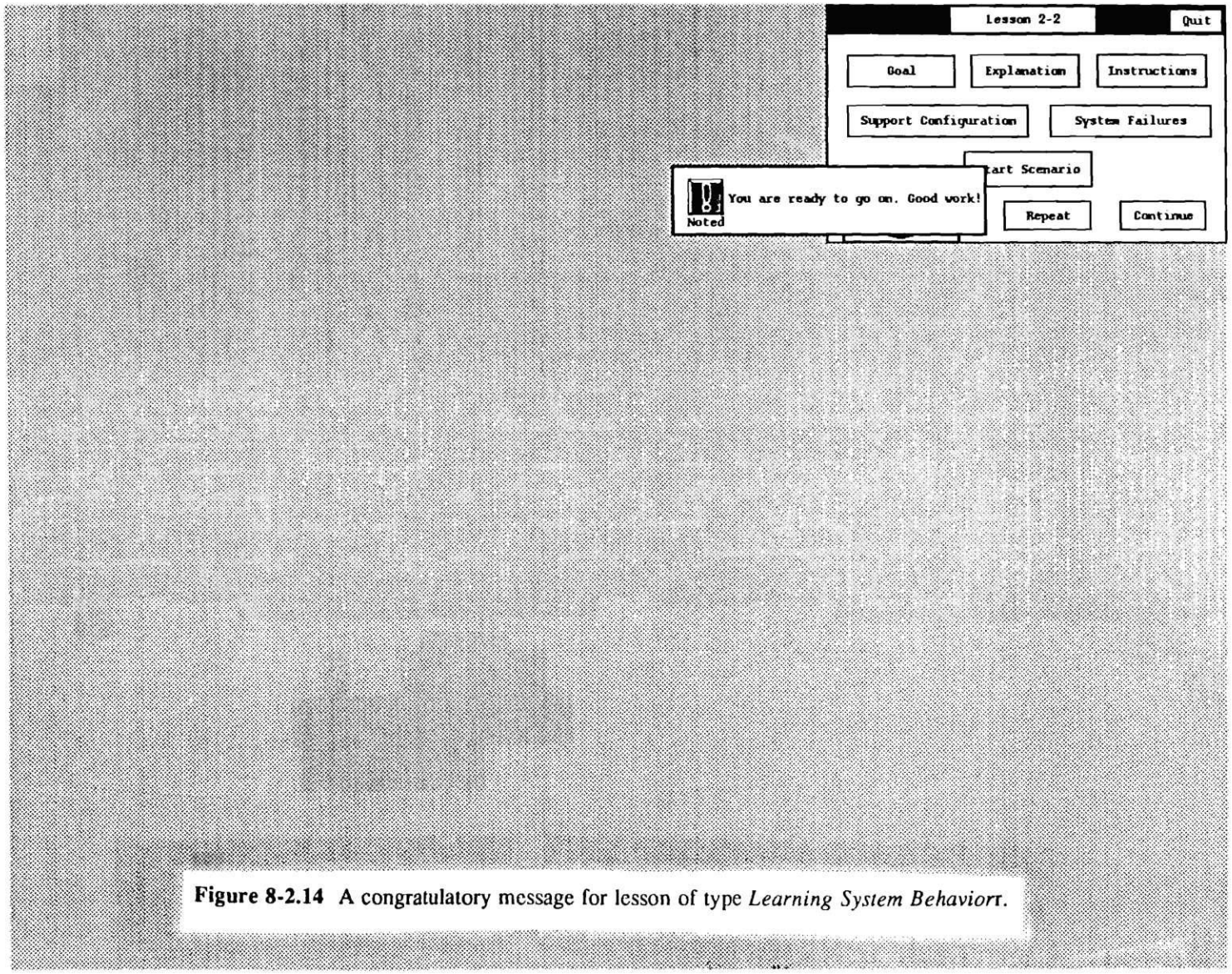


Figure 8-2.14 A congratulatory message for lesson of type *Learning System Behavior*.

student with appropriate messages (Figures 8-2.11, 8-2.12, 8-2.13). The student must resolve these problems before continuing. If all lesson requirements are met, the tutor posts the appropriate congratulatory message as in Figure 8-2.14.

Lesson Type 3: Exploring Tutor's Knowledge

The goal of this lesson is to empower the student with the tutor's conceptual model of the NASA system. The student learns about how the tutor organizes and represent system components, and further explores dynamics of the system and interactions among components.

Example

The unique feature of this lesson is the checkbox at the bottom left corner of the lesson panel labeled "**Inspect Object Representations**" (Figure 8-3.1). The checkbox is a toggle that the student can manipulate to inspect an object's structural representation at any level and at any time during the course of the lesson. When the inspect option is "on", the student can even click on objects that are not clickable otherwise. For example, when the student clicks on the NASCOM object in the Goddard Space Flight Center, the structure and behavior that define a NASCOM object are displayed (Figure 8-3.2). The tutor always displays the object representation panel on the monitor opposite to where the object selected is located. The student has the option to learn more about where the object fits in relation to other objects by selecting the "**Class Hierarchy**" button on the object representation panel. A panel with the same name displays schematically the tutor's hierarchical organization of NASA system components (Figure 8-3.3).

Within the inspect mode, when the student first clicks on an object that is "normally" inspectable, the object's representation is displayed. After closing the object representation panel, the student can click on the object again to inspect the components it comprises. Then, the student can click on any one of the components to view its object structure. For example, the student first clicks on the Command & Data Handling Subsystem object in the Spacecraft Subsystems panel to view the subsystem's structure. Then the student clicks on the object again to bring up the Command & Data Handling Subsystem panel. When

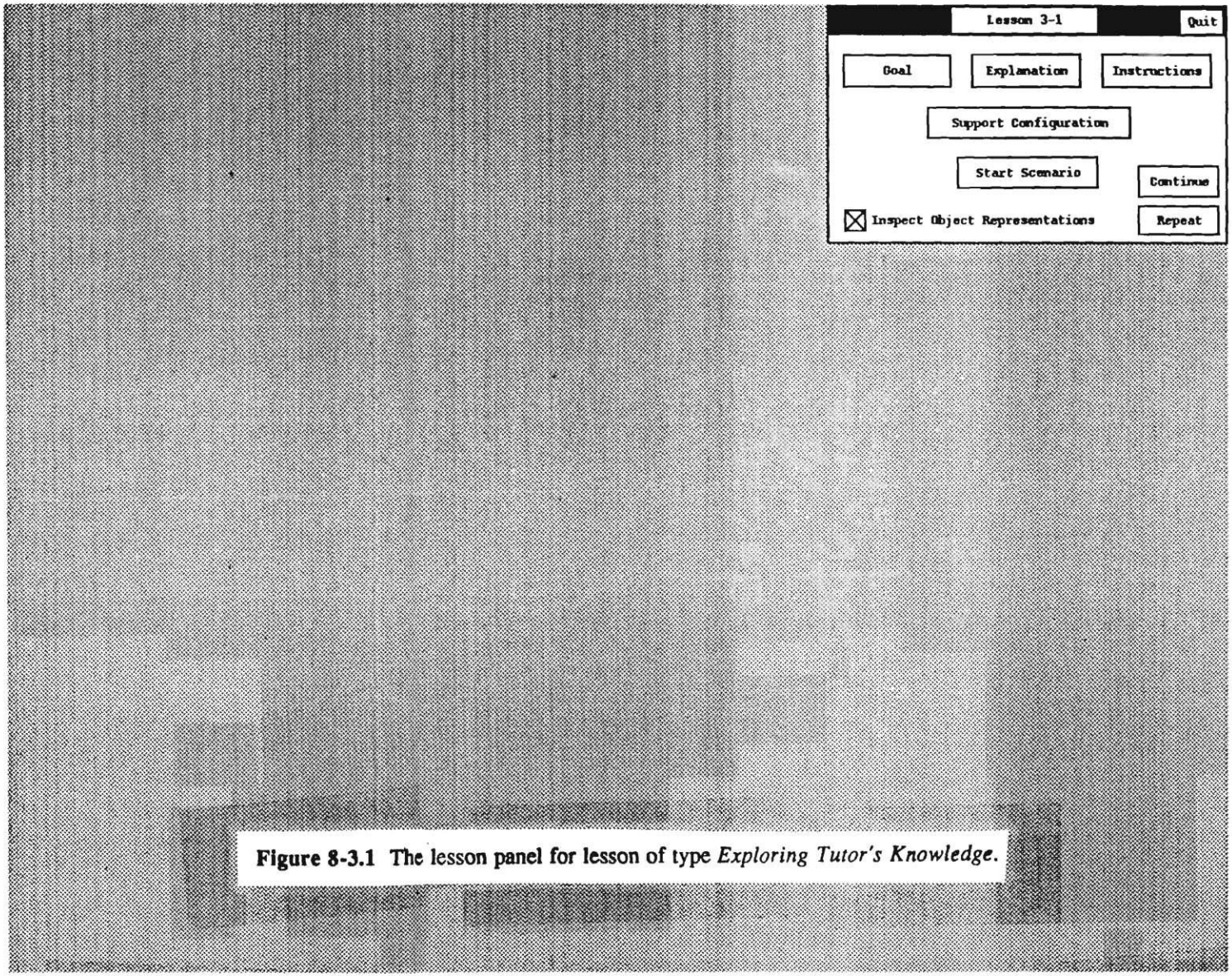


Figure 8-3.1 The lesson panel for lesson of type *Exploring Tutor's Knowledge.*

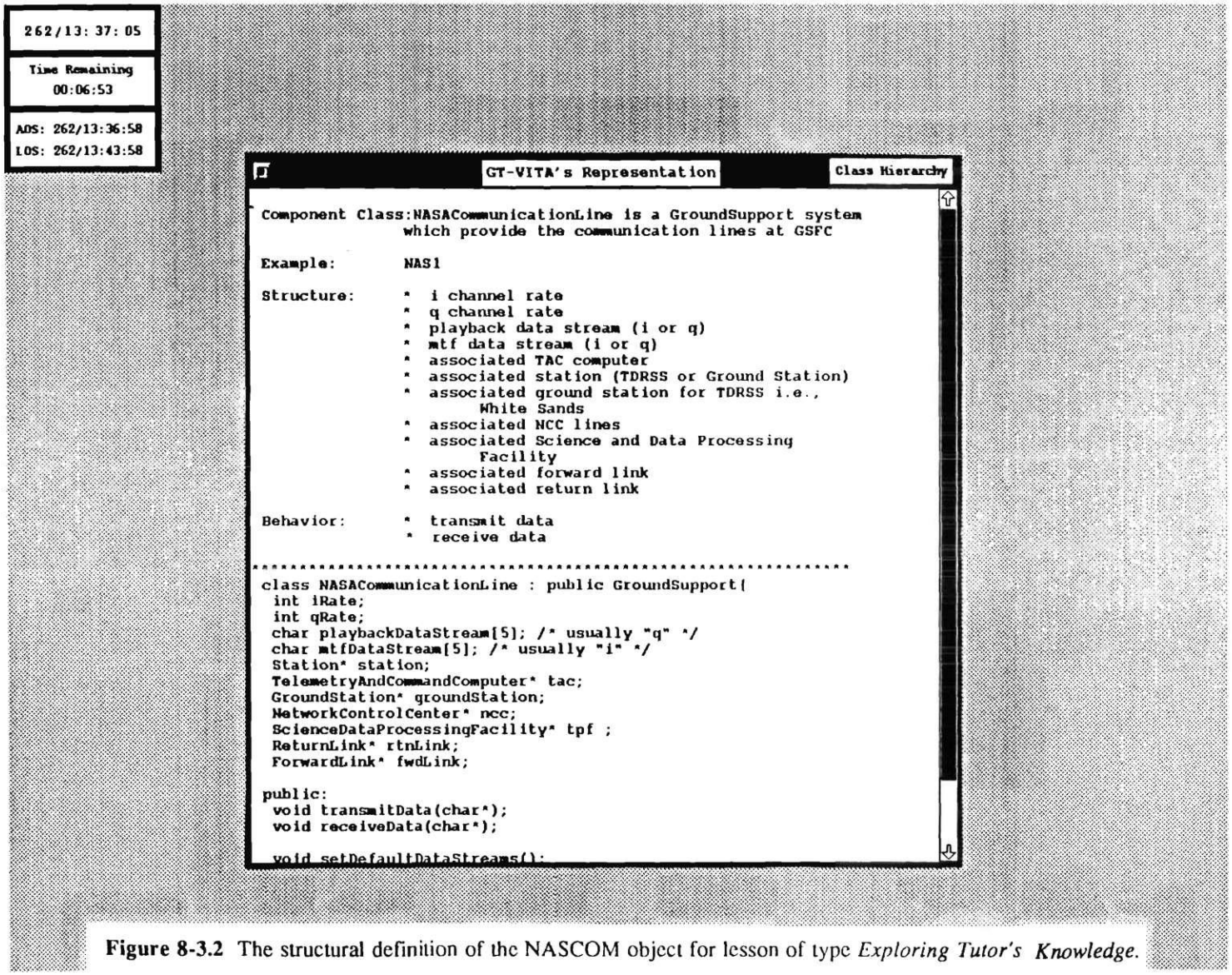


Figure 8-3.2 The structural definition of the NASCOM object for lesson of type *Exploring Tutor's Knowledge.*

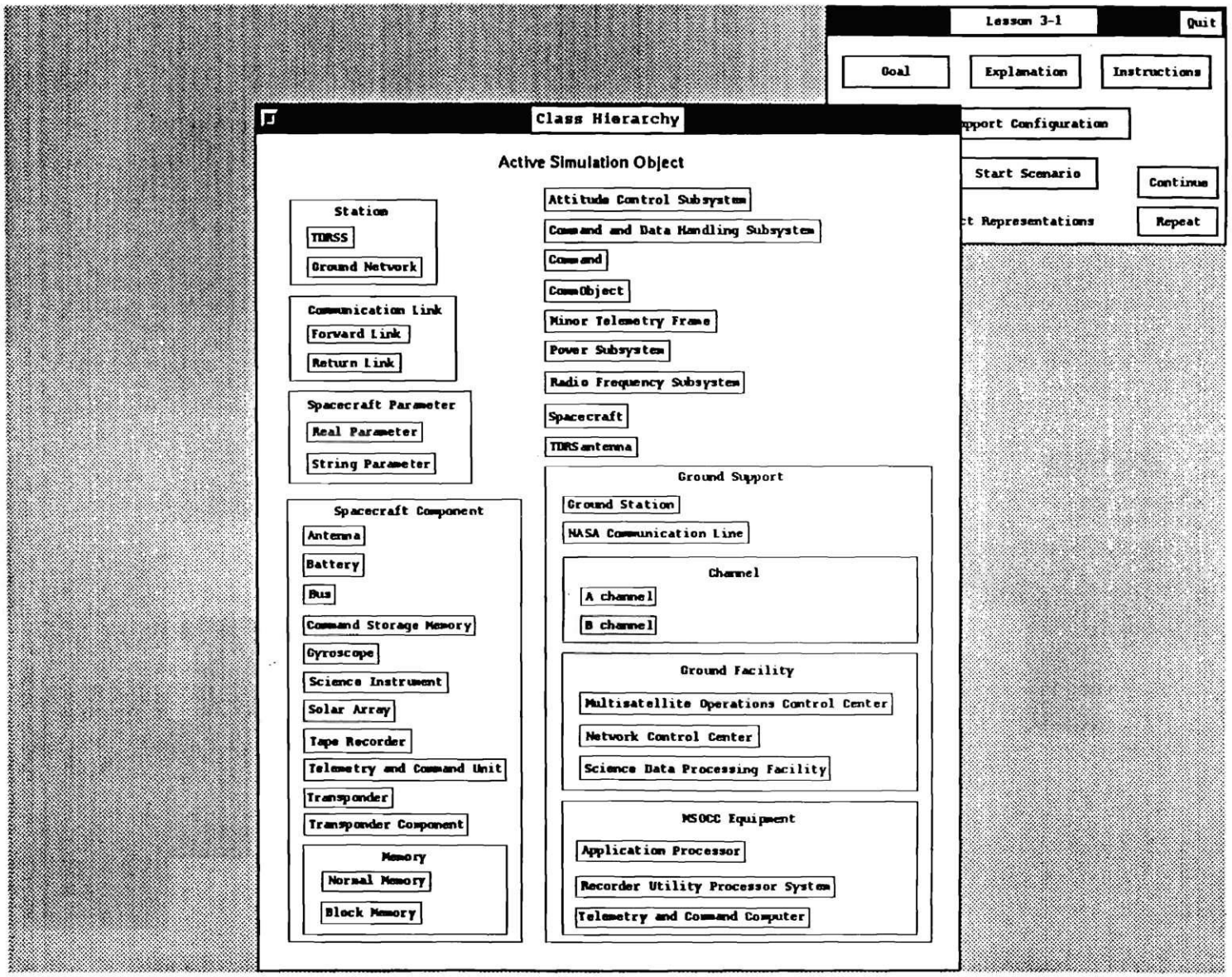


Figure 8-3.3 The GT-POCC object hierarchy for lesson of type *Exploring Tutor's Knowledge*.

the student clicks on the tape recorder object inside this panel, the structure and behavior of a tape recorder is displayed.

When not inspecting object representations, the student watches data flow and system behavior, and becomes more familiarized with the system interfaces. This is the only lesson with no requirements for completion, so the student can end the lesson by selecting "Continue" or "Repeat".

Lesson Type 4: Learning Operations by Example

In this lesson, the student learns about the various operational procedures that a FOT analyst is responsible for in the GT-POCC environment. The tutor plays the role of a FOT analyst and shows the student and executes for the student the commands that are necessary for successful mission control.

Example

After learning about the lesson's general goal, the student selects the "Explanation" button on the lesson panel to find out what the operational procedures the tutor is planning to demonstrate (Figure 8-4.1) For this example, the tutor plans to show pre-pass activities to configure and verify the necessary communications and equipment. The tutor will also demonstrate a spacecraft tape recorder playback activity during the real-time support. The actual set of commands that the tutor has scheduled for these operator activities are displayed when the student clicks on the "**Operator Commands**" button (Figure 8-4.2). The panel with the same name has an "Explain" button that allows the student to get a textual description of the tutor's intentions with regards to the planned commands. The tutor alerts the student with a beep when it is time for a command to be executed. In between commands, the student is encouraged to explore the system and learn more about failures and their effects.

When a scheduled command is ready, the tutor pauses the lesson scenario, highlights the command, and signals the student about it with a beep. To find out how the command is executed, the student selects the "**Action Sequence**" button on the commands panel. A sequence of actions for a command is shown, each action represented by the selection of an object located on a particular panel (Figure 8-4.3). For

example, the first action to accomplish the command "AP1 init" is to select the Goddard Space Flight Center object on the NASA Data/Information System panel. The second action is to select the MSOCC object on the Goddard Space Flight Center panel and so on. To find out more about the command, the student clicks on the "Explain" button on the Action Sequence panel which displays a textual description of the command. To watch the tutor take on the actions, the student clicks successively on the button "**Step**". The action is highlighted at each step. When an object selection involves a choice of options, the tutor highlights the option appropriate for the current command. Figure 8-4.4 shows the sequence of panels that resulted from the tutor's actions for "AP1 init". To view alternative action sequences for the same command, the student clicks on the "**Other(s)**" button. Figure 8-4.5 shows the another action sequence for "AP1 init".

Instead of stepping through each action, the student has the option to "**Execute All**" the actions for the command from either the Operator Commands or Action Sequence panels. In this way, the student sets the pace and style of learning command execution.

After the entire action sequence has been completed (signaled by a beep), the tutor waits for the student to restart the lesson scenario. When the student is ready, the student selects "Continue" from either the Operator Commands or Action Sequence panels to restart the lesson scenario. At this time, the student watches the effects of the command just executed on system behavior. In the current example, the student closes the control panel for the Application Processor and observes that the border of the Application Processor object turns from white (idle) to green (initialized). On the Operator Commands Panel, the tutor marks the "AP1 init" command as completed with three asterisks in the column labeled "Done" (Figure 8-4.6. While waiting for the next scheduled command, the student can freely explore the system interfaces.

As the tutor demonstrates a tape recorder playback activity, the student watches the spacecraft command (issued by the tutor) transmit from the ground to the tape recorder. The state of the tape recorder dynamically changes, and playback data transmits from the tape recorder to the ground, specifically to the Recorder Utility Processor System in the MSOCC facility and to the Science Data Processing Facility.

The lesson ends when the set of scheduled operator commands have been accomplished. As in other lessons, the tutor informs the student about it and the student may decide to continue or repeat the lesson.

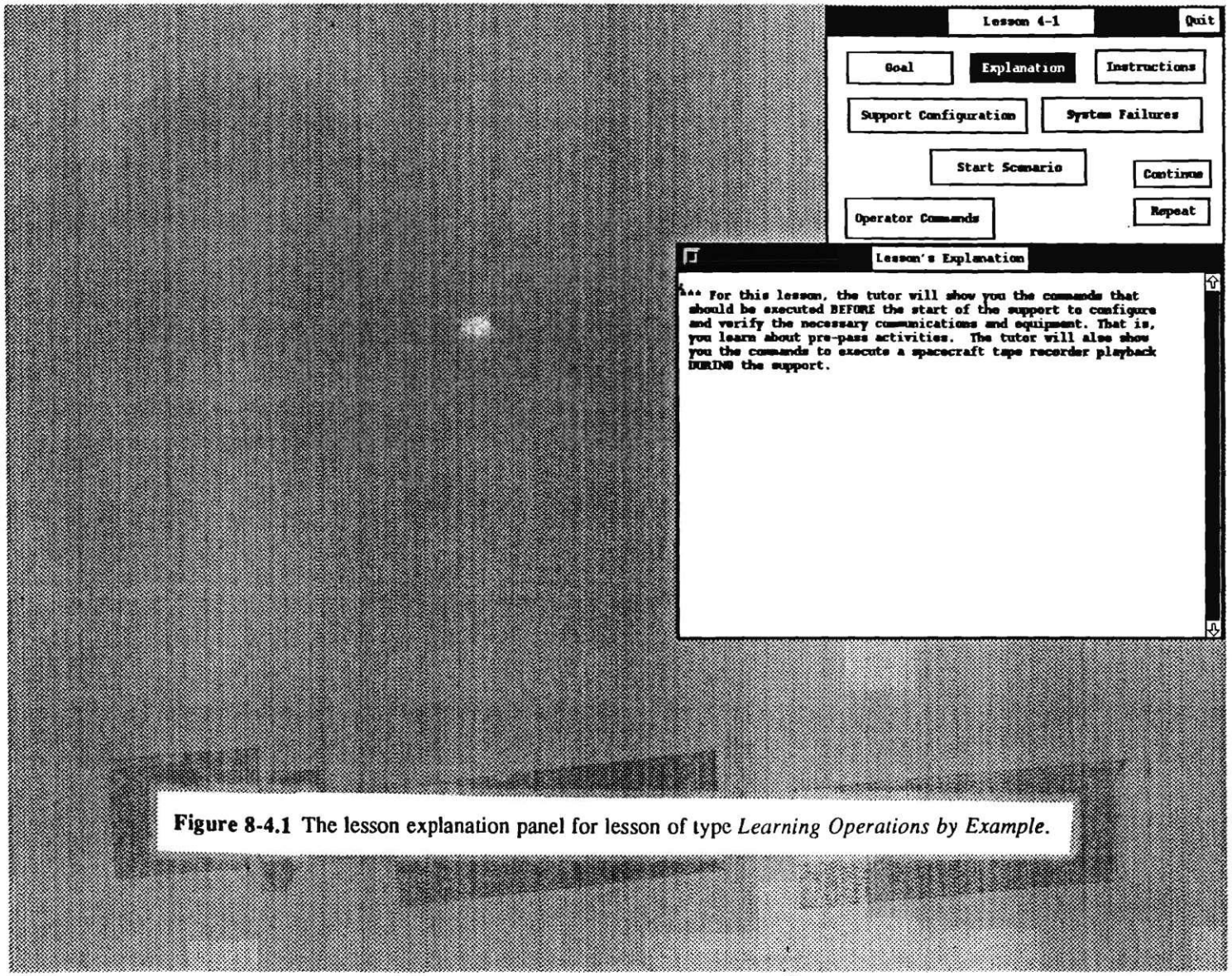


Figure 8-4.1 The lesson explanation panel for lesson of type *Learning Operations by Example.*

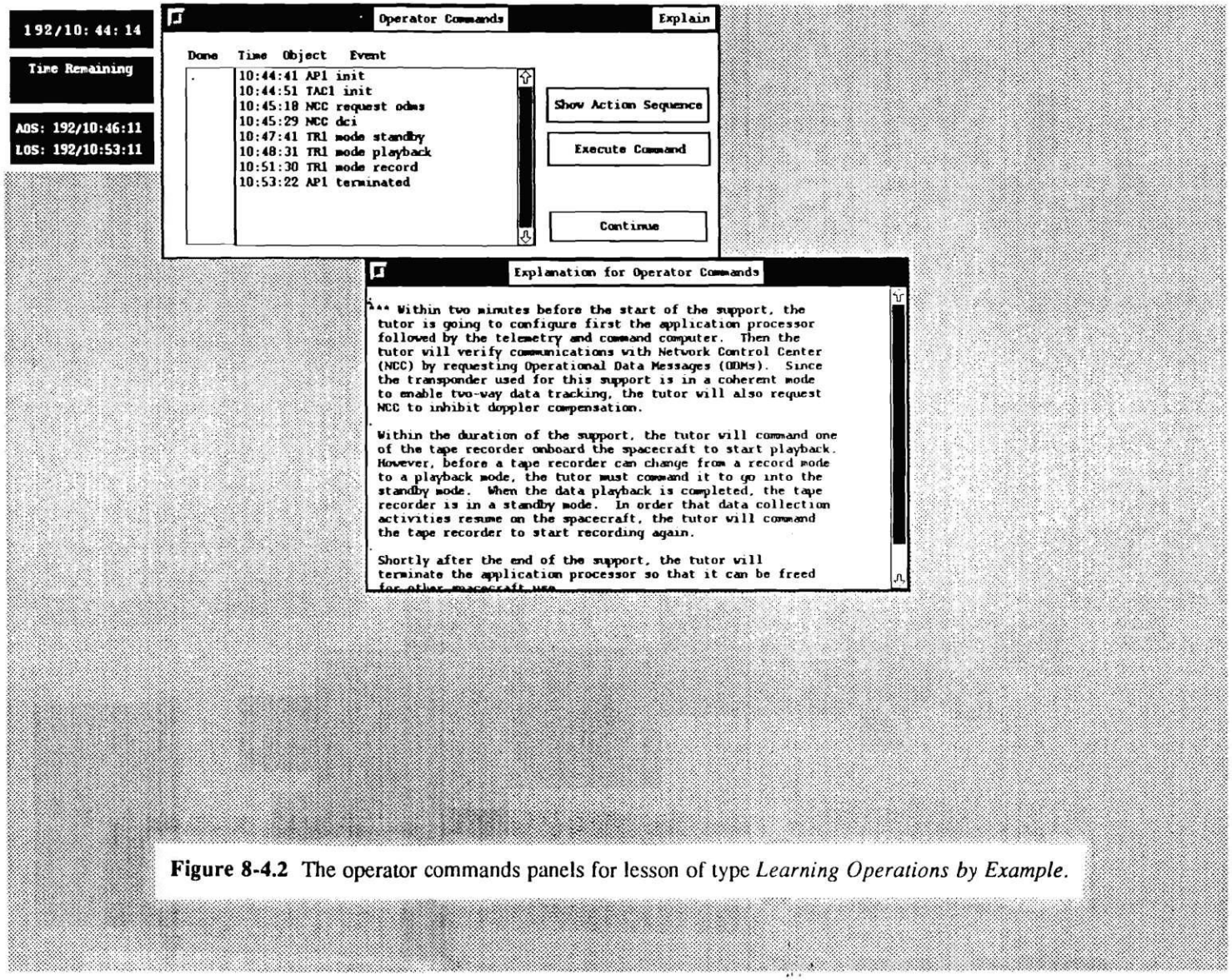


Figure 8-4.2 The operator commands panels for lesson of type *Learning Operations by Example.*

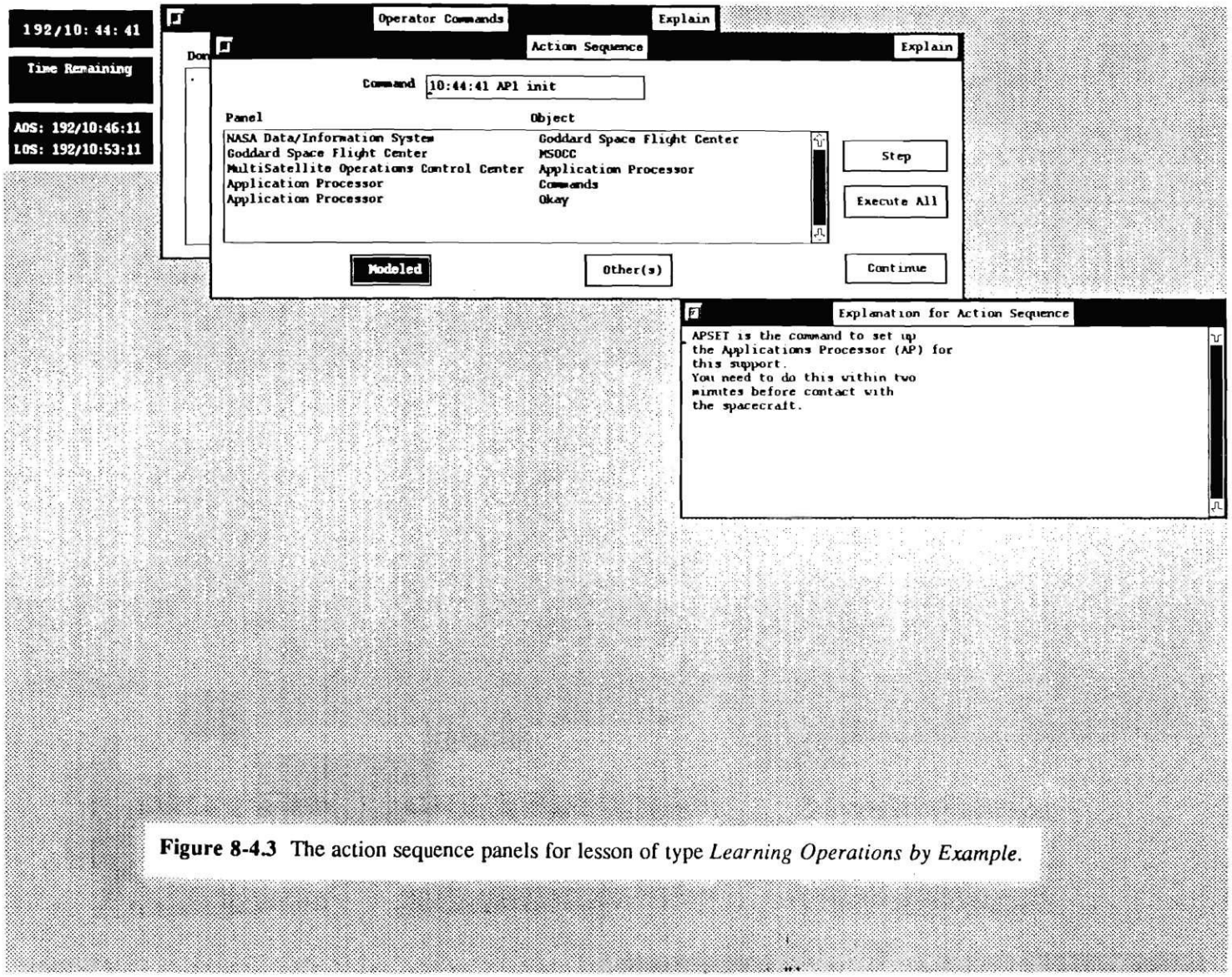


Figure 8-4.3 The action sequence panels for lesson of type *Learning Operations by Example.*

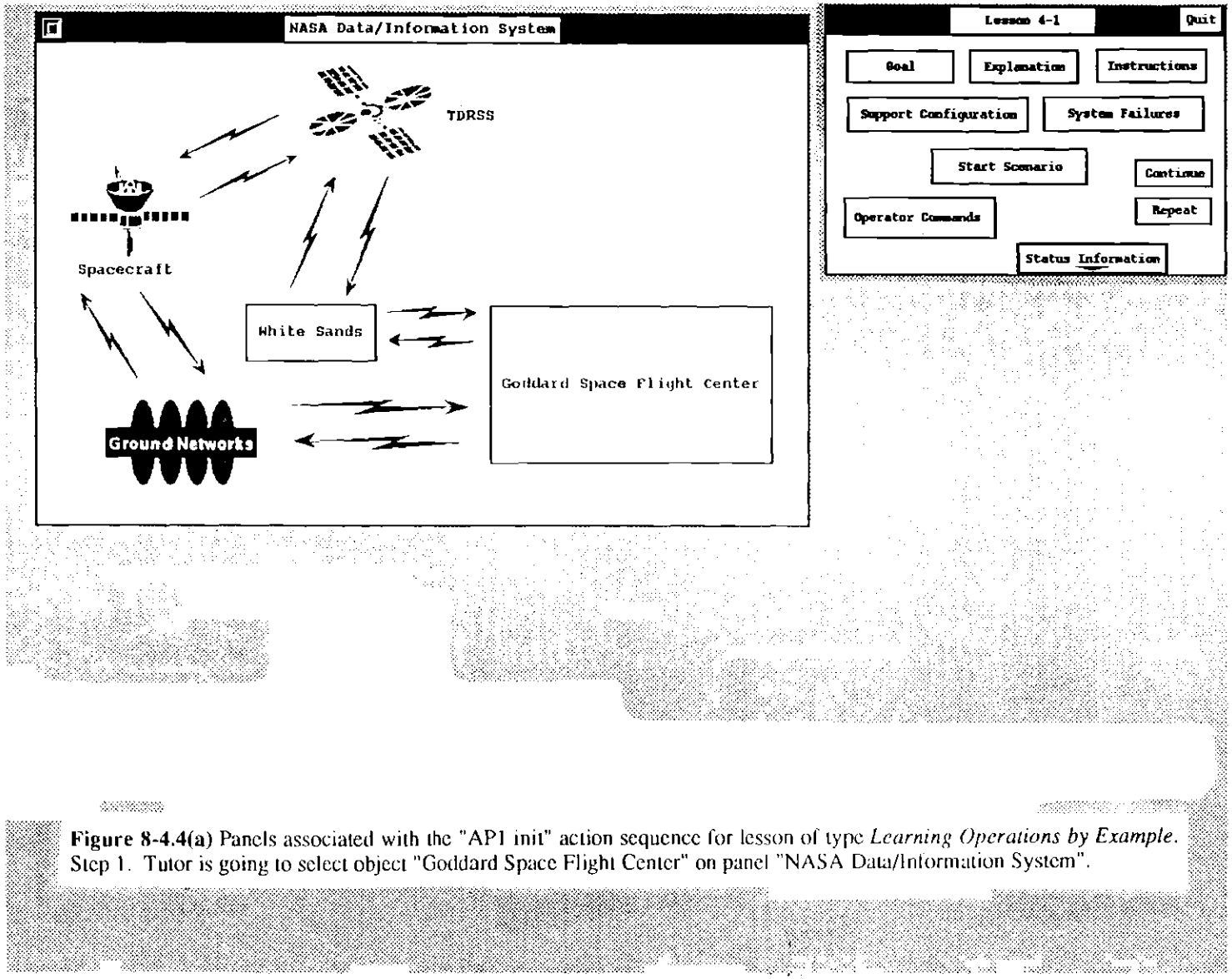


Figure 8-4.4(a) Panels associated with the "AP1 init" action sequence for lesson of type *Learning Operations by Example*. Step 1. Tutor is going to select object "Goddard Space Flight Center" on panel "NASA Data/Information System".

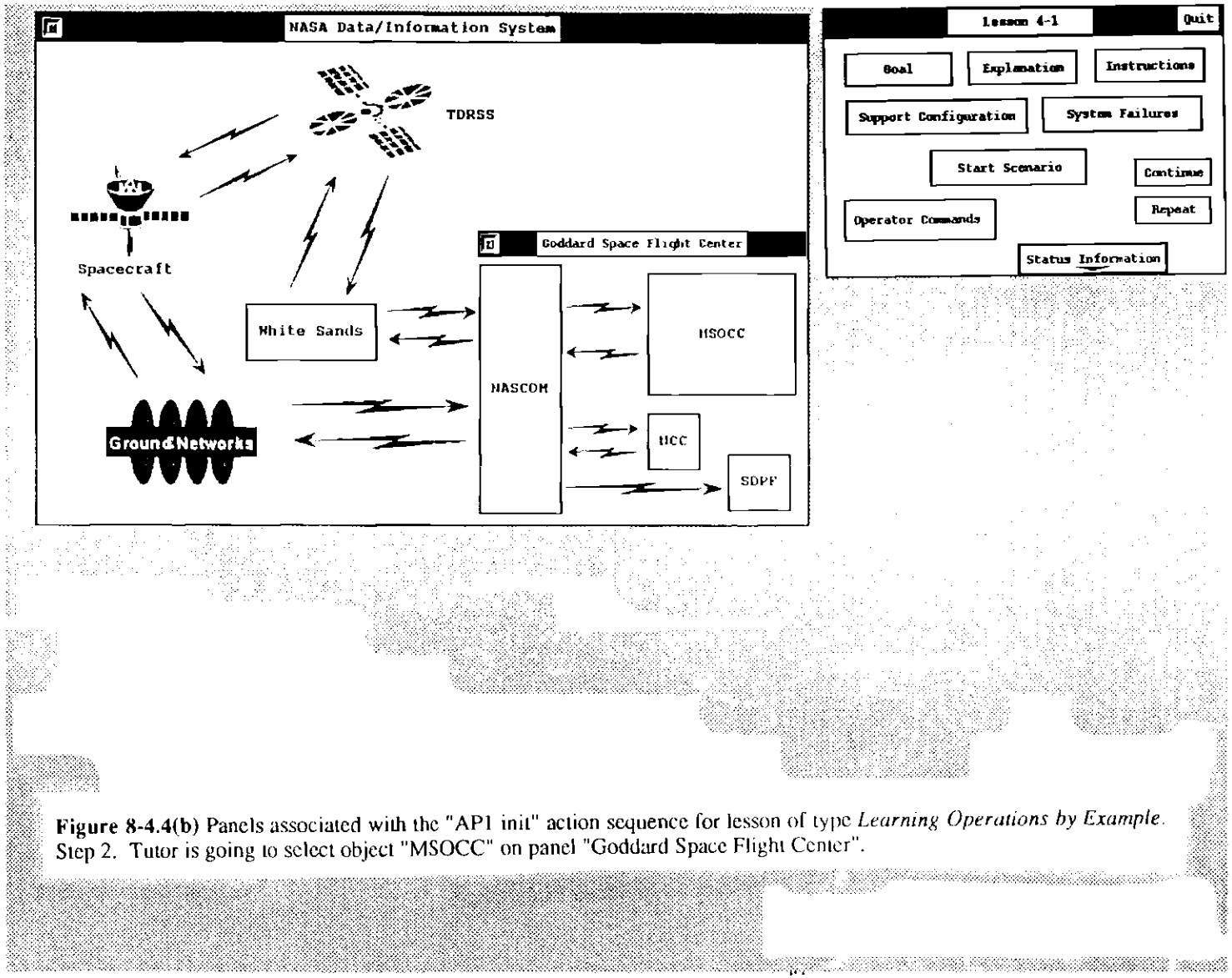


Figure 8-4.4(b) Panels associated with the "AP1 init" action sequence for lesson of type *Learning Operations by Example*. Step 2. Tutor is going to select object "MSOCC" on panel "Goddard Space Flight Center".

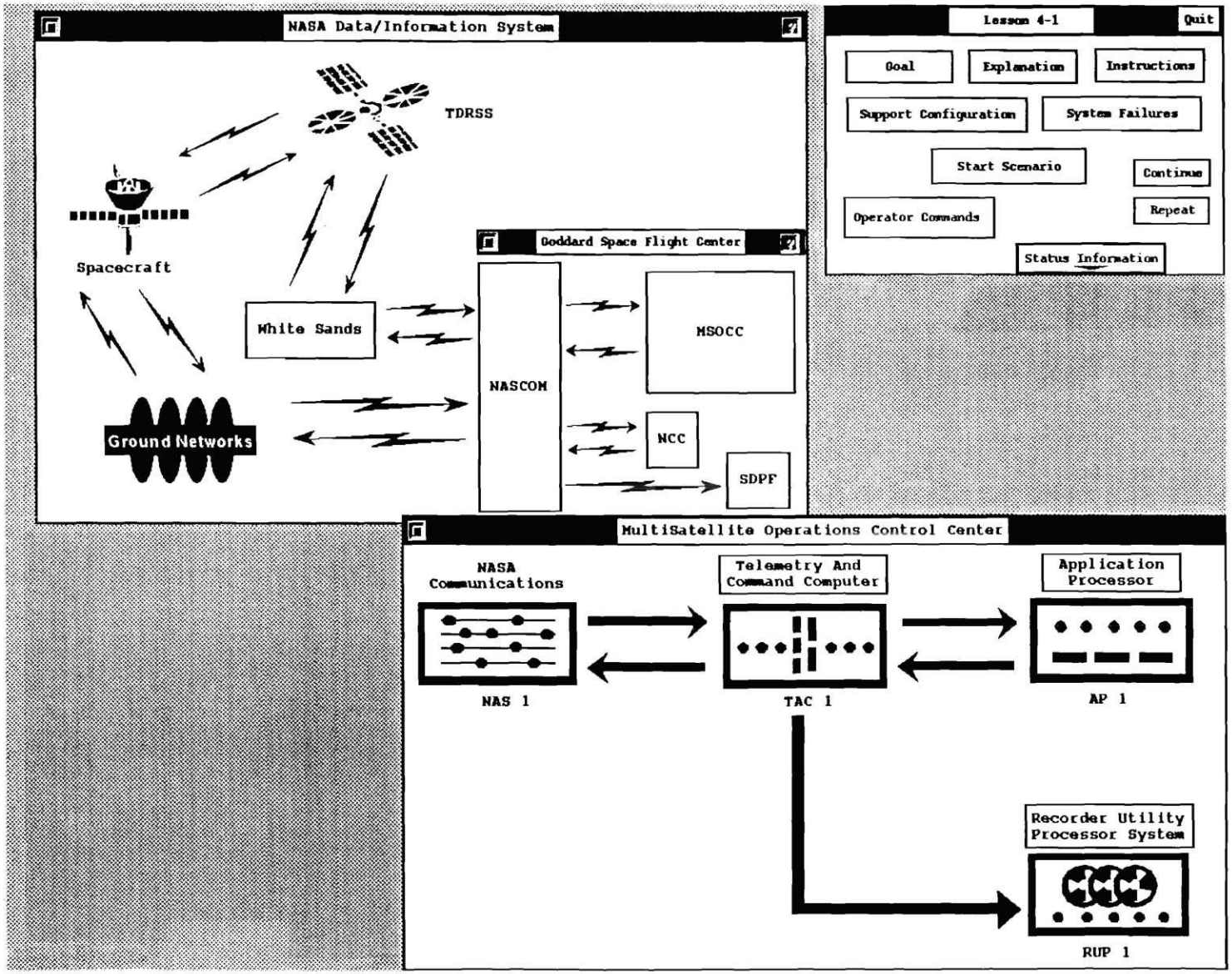


Figure 8-4.4(c) Panels associated with the "AP1 init" action sequence for lesson of type *Learning Operations by Example*. Step 3. Tutor is going to select object "Application Processor" on panel "Multisatellite Operations Control Center".

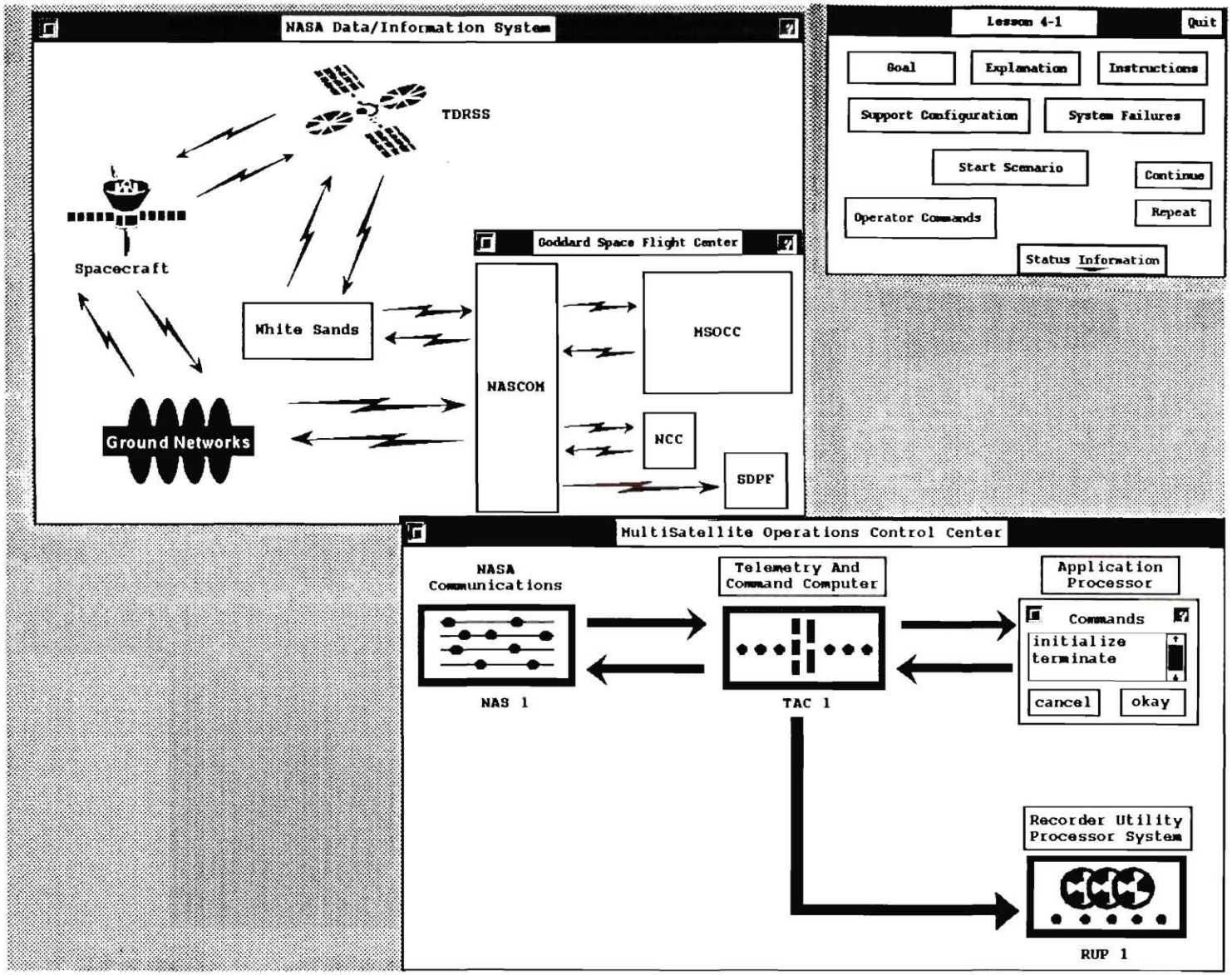


Figure 8-4.4(d) Panels associated with the "AP1 init" action sequence for lesson of type *Learning Operations by Example*. Step 4. Tutor is going to select option "initialize" of object "commands" on panel "Application Processor".

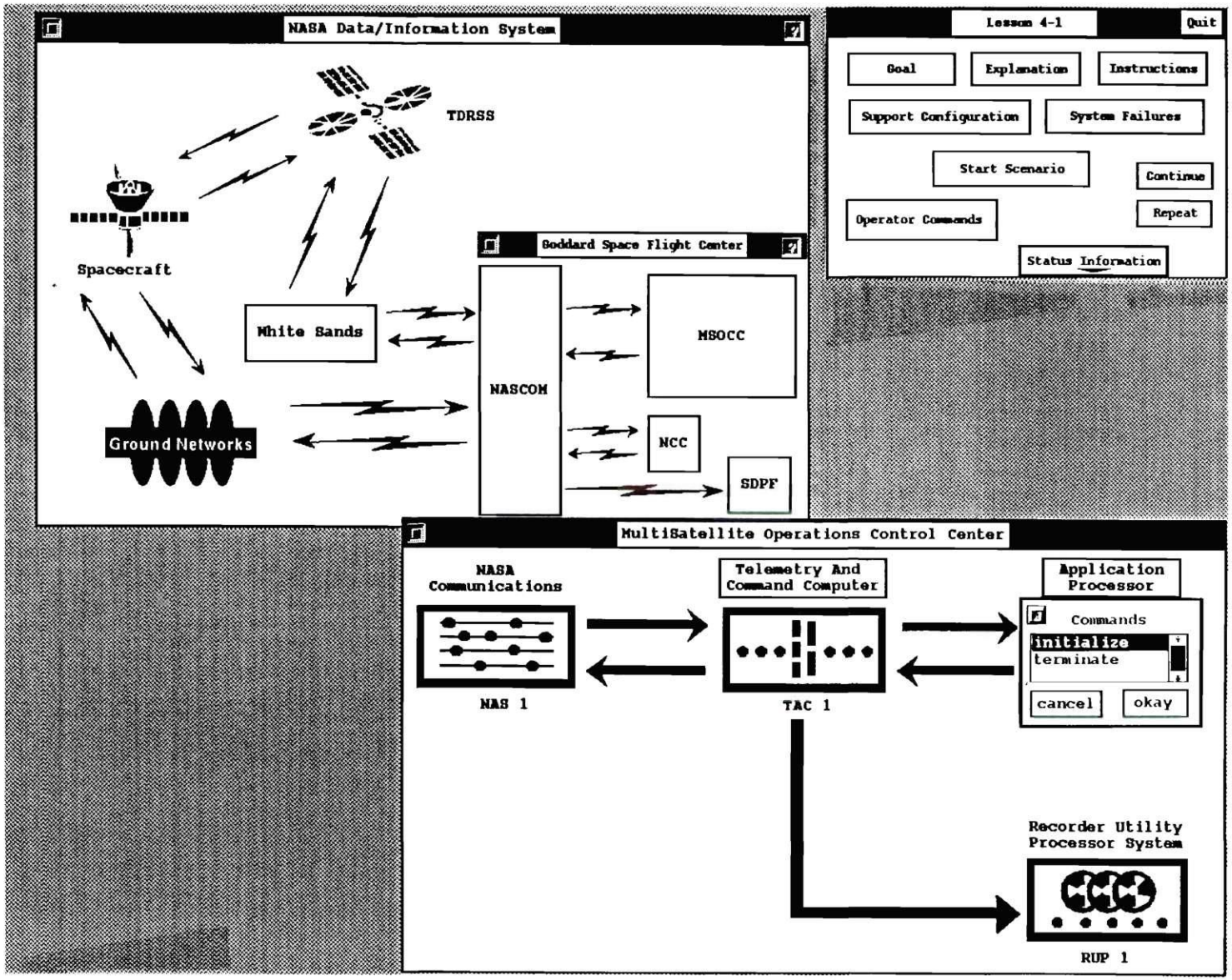


Figure 8-4.4(e) Panels associated with the "AP1 init" action sequence for lesson of type *Learning Operations by Example*. Step 5. Tutor is going to end action sequence by selecting the "okay" button..

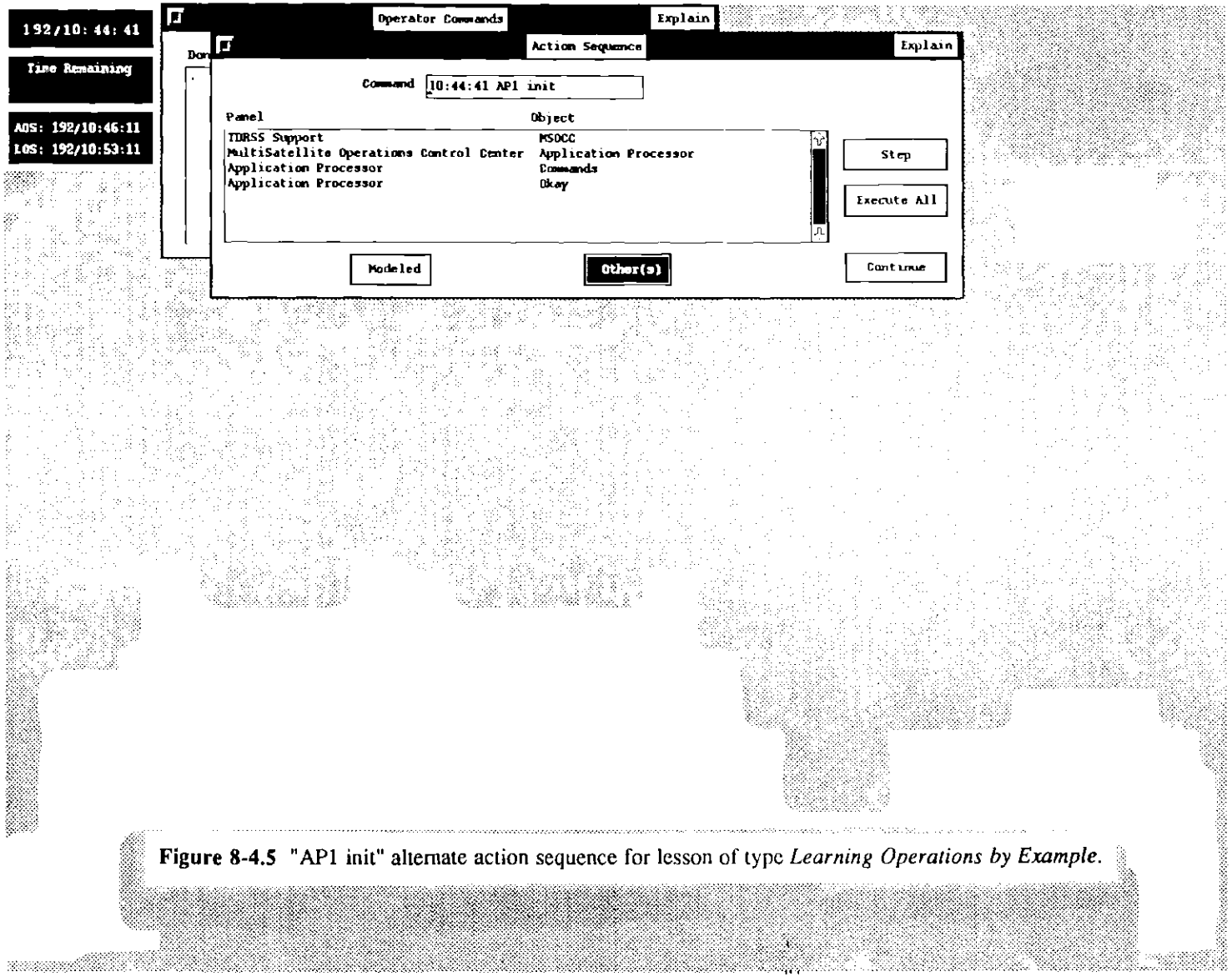


Figure 8-4.5 "AP1 init" alternate action sequence for lesson of type *Learning Operations by Example.*

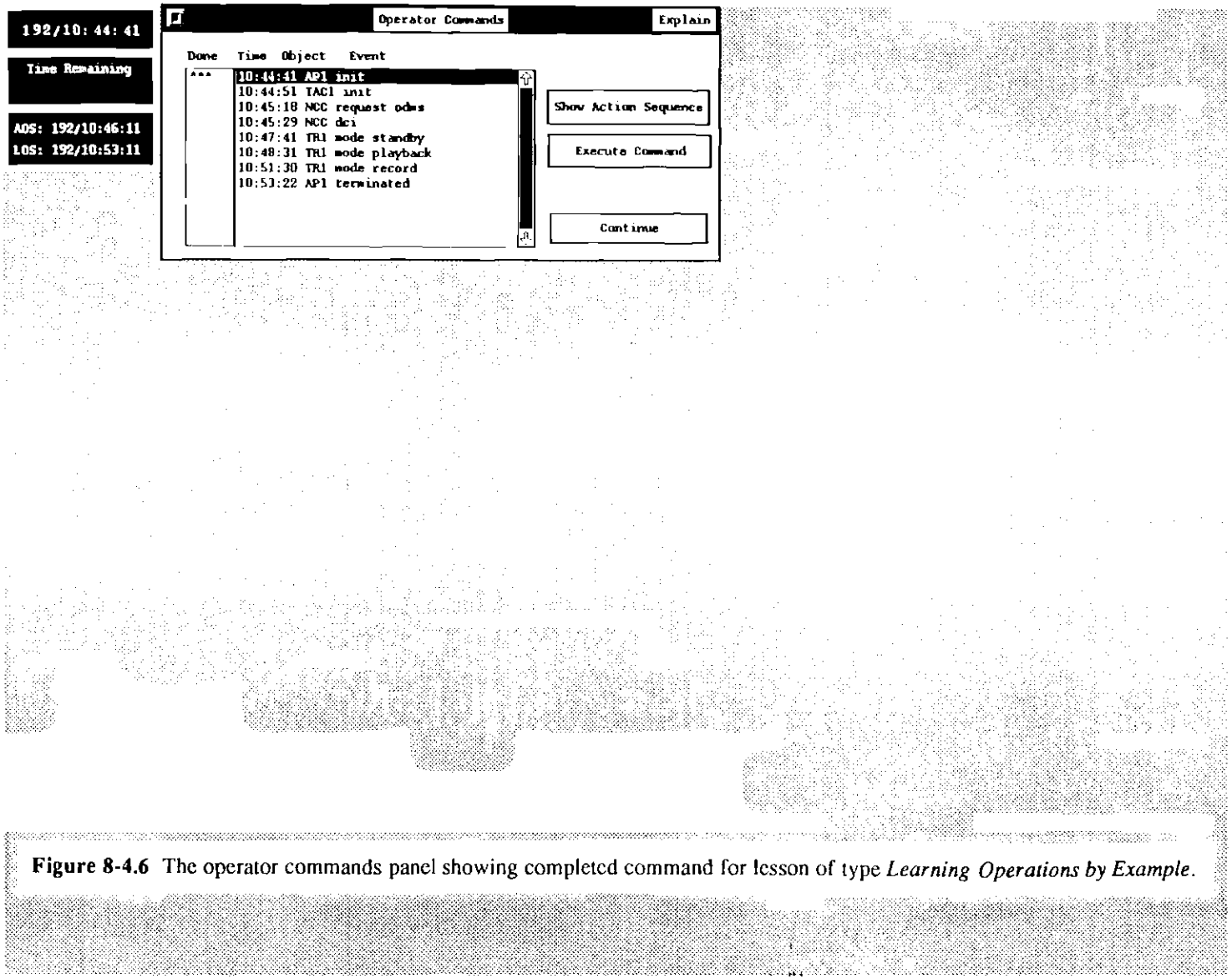


Figure 8-4.6 The operator commands panel showing completed command for lesson of type *Learning Operations by Example*.

Lesson Type 5: Learning Operations by Doing

In this lesson, the student learns to execute pre-pass activities to configure, verify and setup the necessary communications and equipment before the start of the support.

Example

The lesson's goal is shown in Figure 8-5.1 The tutor lists out the three major pre-pass operator functions: configuration, verification and setup. For each function, the tutor also specifies the tasks to achieve each function. This decomposition of pre-pass activities is consistent with the underlying operator function model of GT-POCC (Refer to Table 6-2). The tutor instructs one task at a time, and the description for the current task is obtained through the "**Task Explanation**" button. The dynamics of this lesson type is similar to the *Learning Operations by Example* lesson type. The tutor alerts the student when it is time to perform a task. The tutor pauses the scenario while the student learns to execute the actions associated with the task. The student resumes the scenario to watch the effects of the task just performed. Again, in between task instruction, the student can freely explore the system interfaces.

For example, the tutor alerts the student about a task called V_NCC (Figure 8-5.2). The student gets the Task Explanation panel by selecting the button with the same name. This panel shows a description of the task in terms of what it is, when and why it must be done (Figure 8-5.3). Next, the student finds out about the steps in performing the V_NCC task by selecting the "**Steps in Task**" button. The new panel shows a checklist of commands to accomplish the task (Figure 8-5.4) It also shows the current step that should be taken. To take the current step, the student clicks the "**Help**" checkbox for "NCC request odms" which displays the action sequence and description of the command selected (Figure 8-5.5). The format of this panel is consistent with that from the *Learning Operations by Example* lesson type. That is, the action sequence specifies the object to be selected on a panel and the option value to be selected if the object provides a choice. Based on this help panel, the student proceeds to undertake each step in the action sequence as suggested. The tutor posts an error message if the student's action does not match any in the

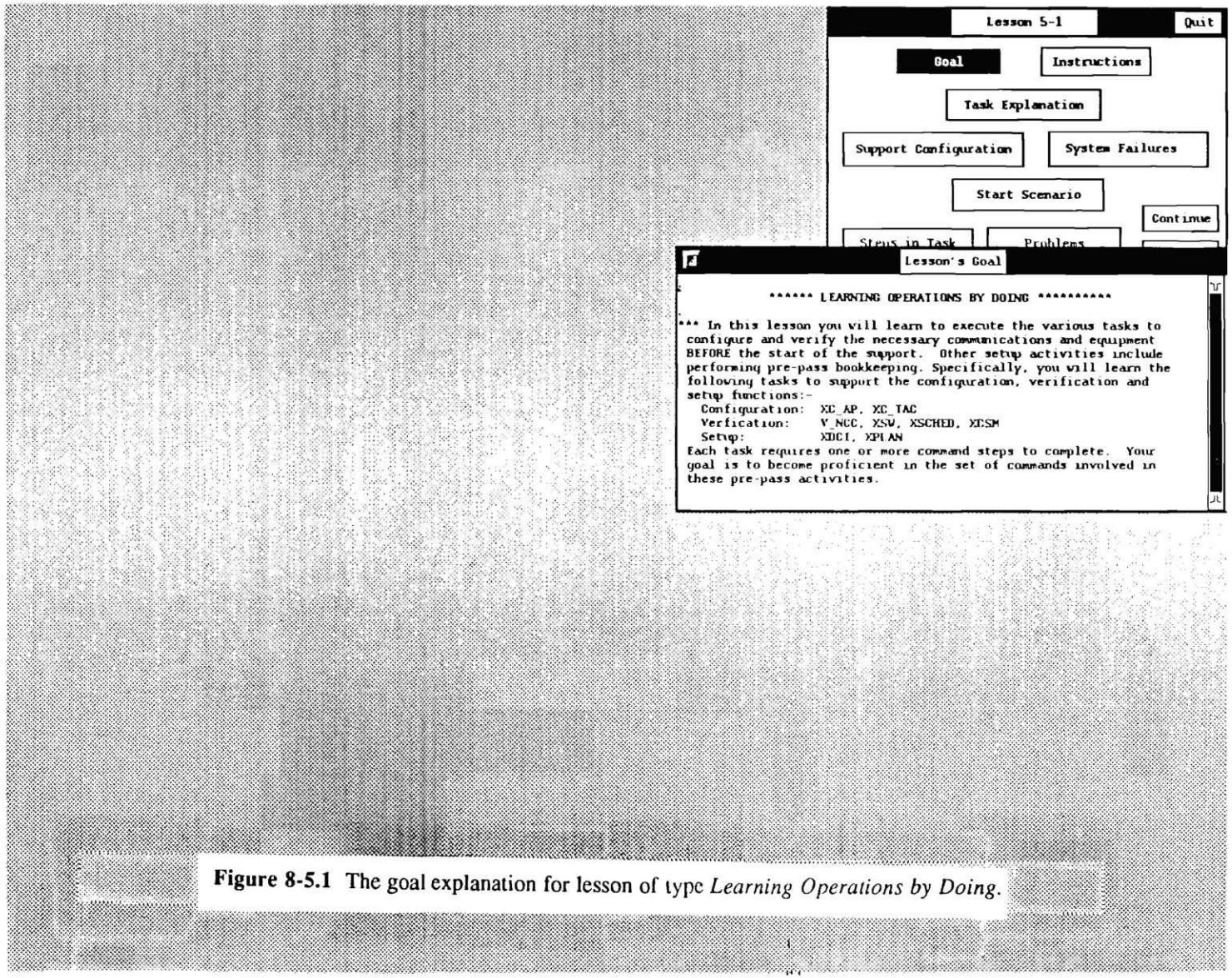


Figure 8-5.1 The goal explanation for lesson of type *Learning Operations by Doing*.

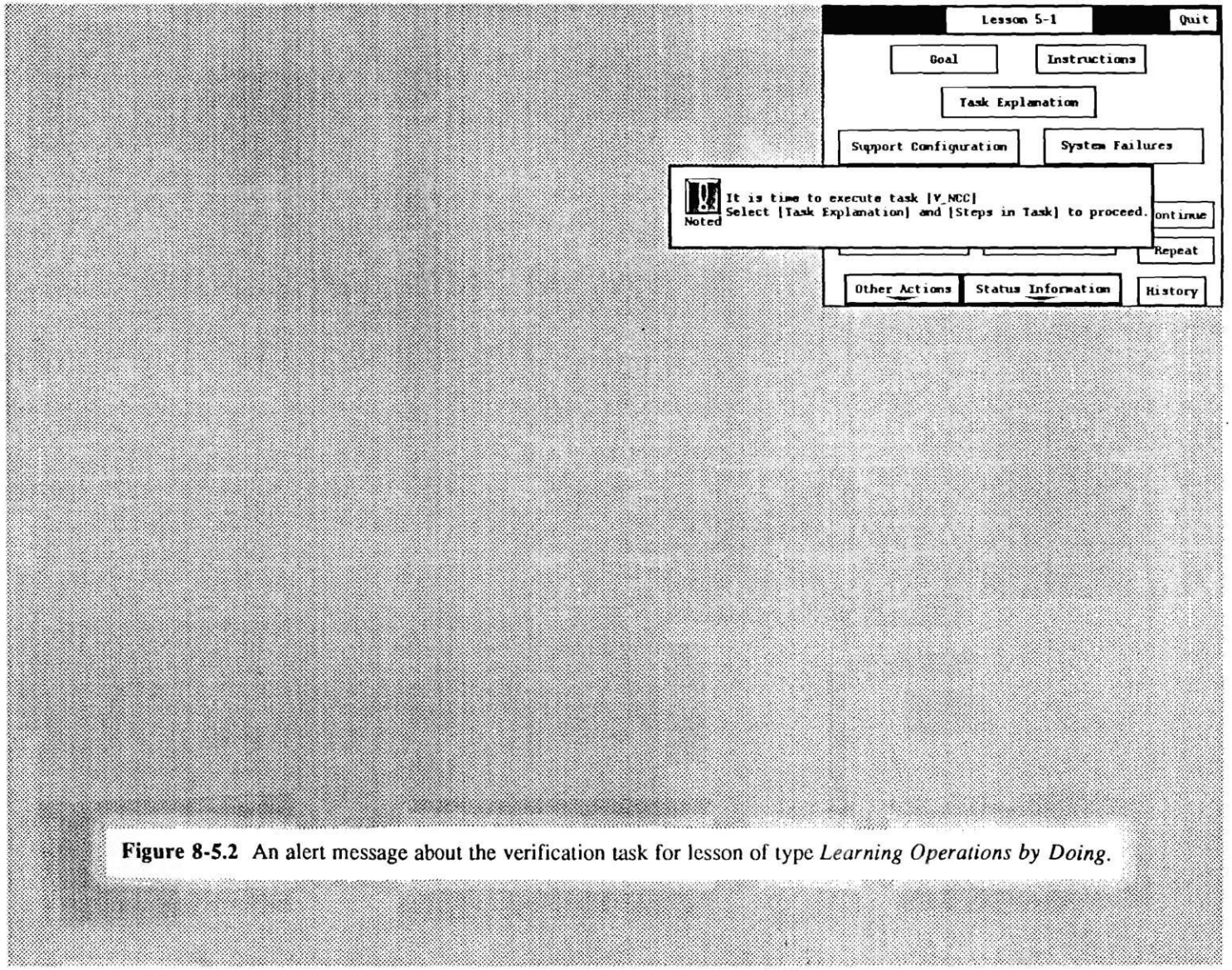


Figure 8-5.2 An alert message about the verification task for lesson of type *Learning Operations by Doing.*

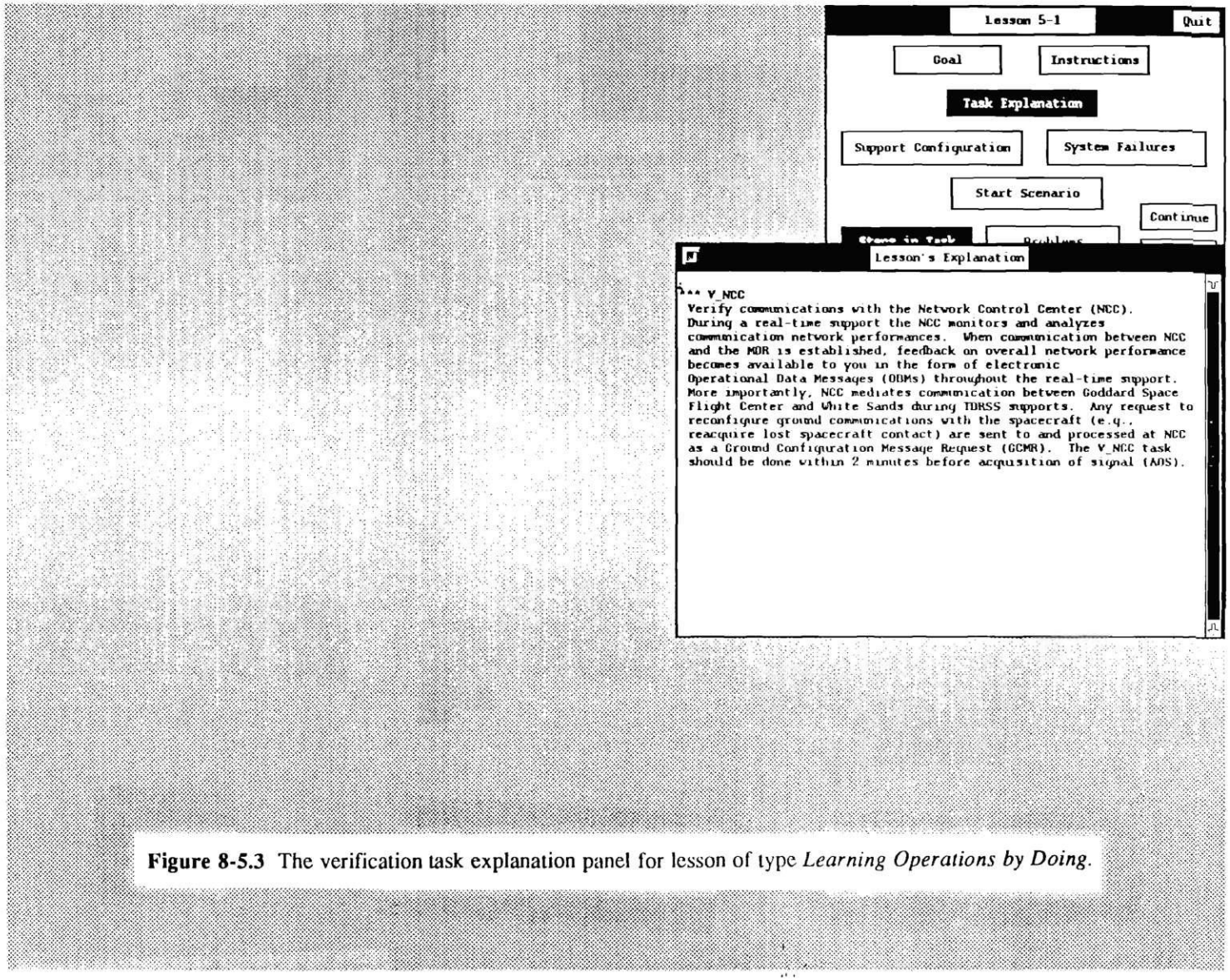


Figure 8-5.3 The verification task explanation panel for lesson of type *Learning Operations by Doing*.

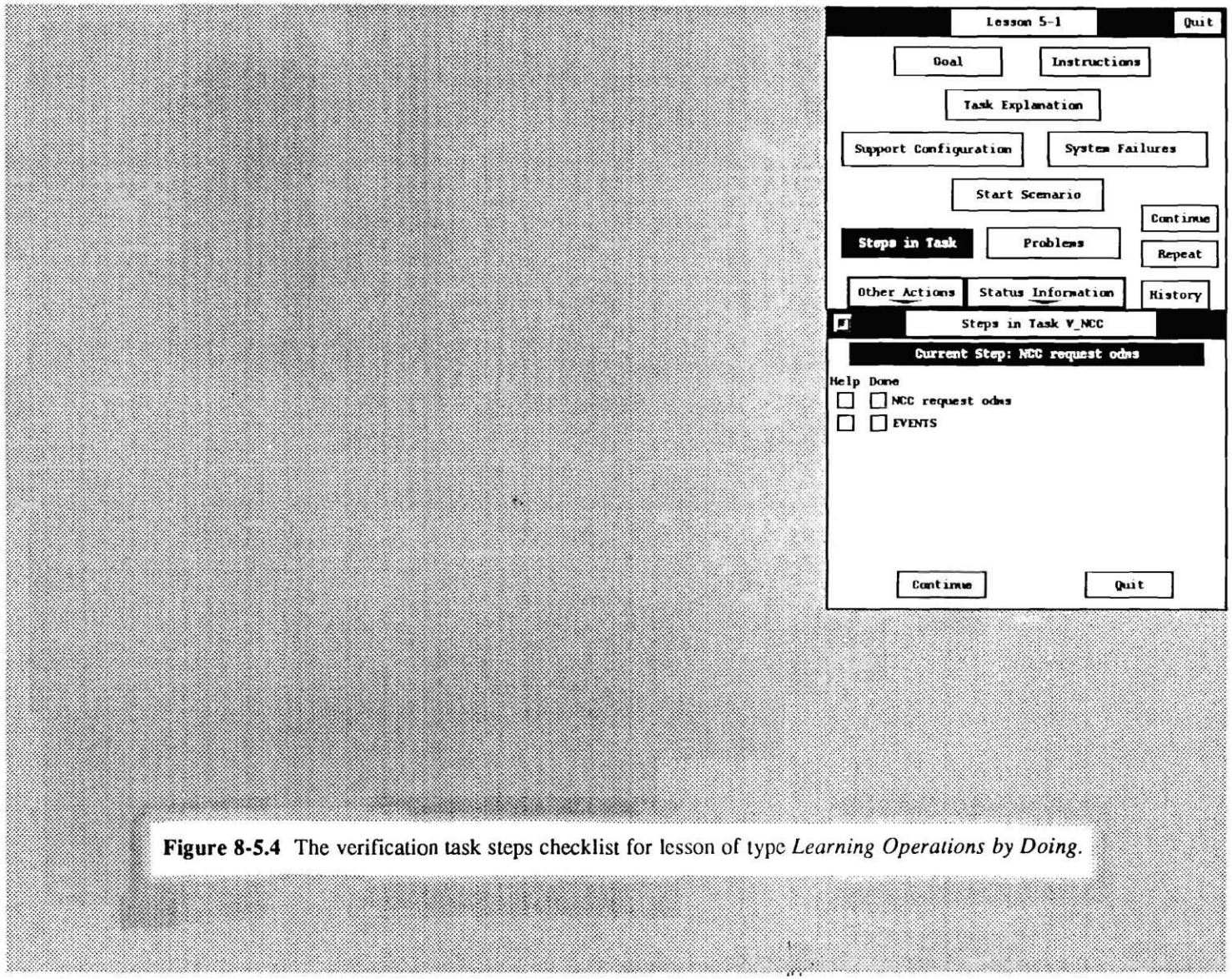


Figure 8-5.4 The verification task steps checklist for lesson of type *Learning Operations by Doing.*

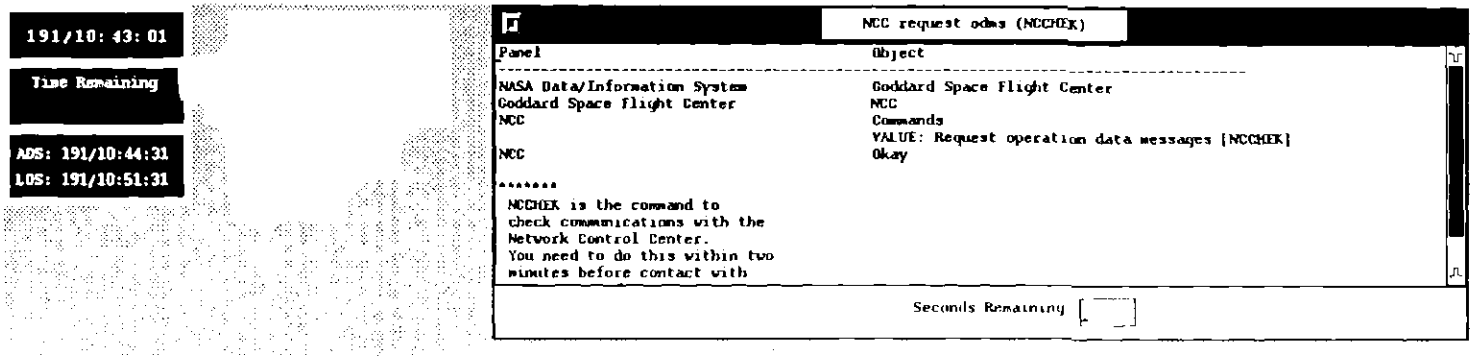


Figure 8-5.5 The verification task "NCC request odms" action sequence for lesson of type *Learning Operations by Doing*.

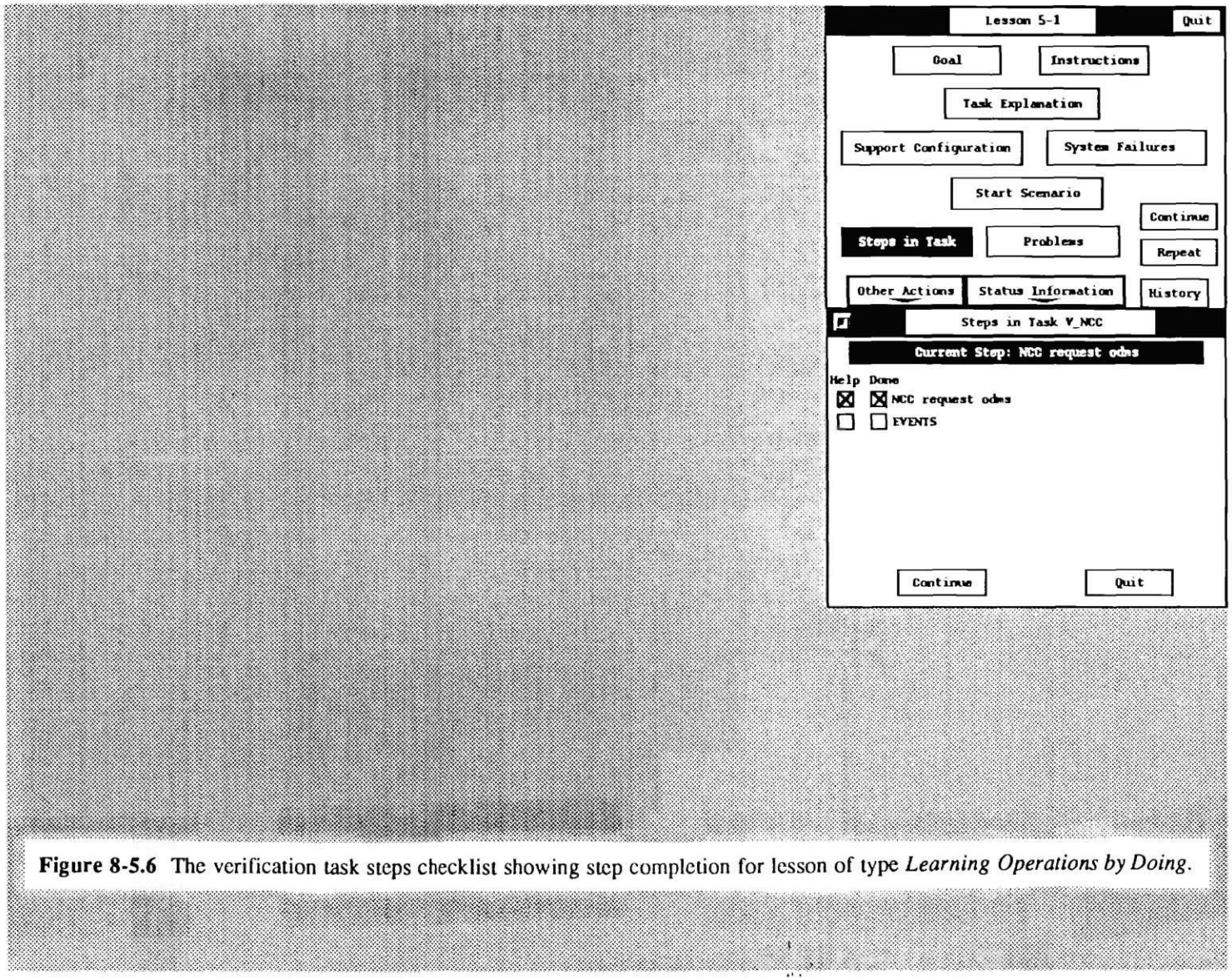


Figure 8-5.6 The verification task steps checklist showing step completion for lesson of type *Learning Operations by Doing*.

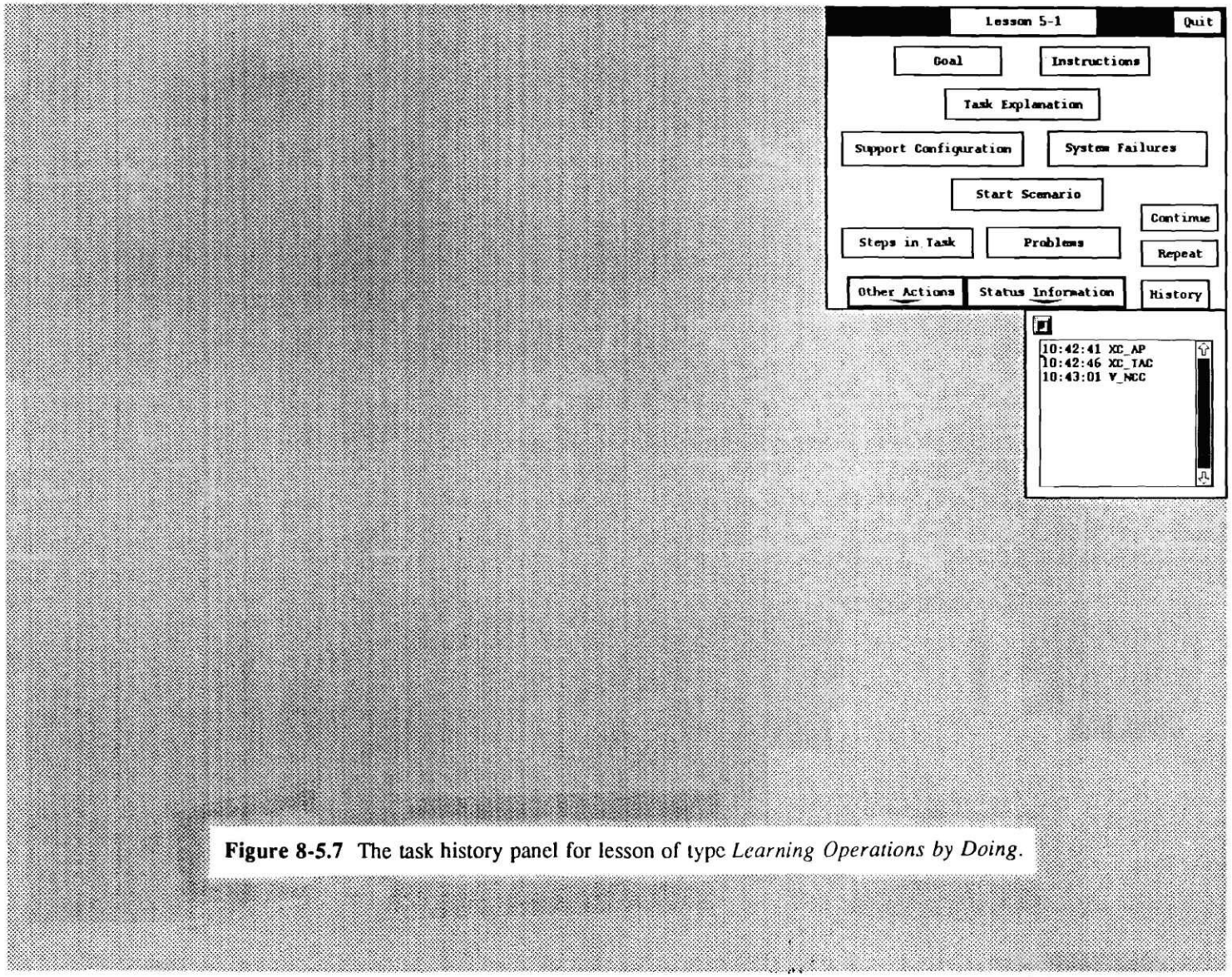


Figure 8-5.7 The task history panel for lesson of type *Learning Operations by Doing*.

sequence. When a command is completed, the tutor marks it as "**Done**", as shown in Figure 8-5.6). The student is required to complete all steps in the checklist before continuing. The student selects "Continue" either on the Steps in Task or lesson panel to restart the lesson scenario and to watch the effects of the task just completed on system behavior.

As each task is completed, the tutor records it in a time-stamped list. The student selects "**History**" on the lesson panel to display the list of tasks learned so far (Figure 8-5.7). The final task history should match the tasks specified in the Lesson's Goal panel. Often, the tasks to be instructed are achieved before the lesson scenario ends, as in this example, where pre-pass activities are taught. To reinforce learning, the tutor requires that the student learns on his or her own until the real-time support ends.

Lesson Type 6: Practicing Operations with Feedback

The goal of this lesson is to allow the student to perform the duties of a FOT analyst in a real-time practice environment. The student initiates all pre-pass, on-pass and post-pass activities that are necessary for successful mission control. Meanwhile, the tutor assesses the student's actions throughout the support. If an assessment is unsatisfactory, the tutor provides immediate feedback and information for rectifying the problem.

Example

Figure 8-6.1 shows the lesson panel for this sample lesson. The "System Failures" button is appropriately absent for all practice lessons. All control actions and information requests not represented through the domain objects are made available with the "**Other Actions**" and "**Status Information**" pull-down menus (see Figures 7-26 to 7-29). The tutor provides immediate feedback with the "**ALERT**" button. The button turns from grey to red, and the tutor alerts with a beep. Once the student starts the scenario, the student must begin to perform various operational procedures in real time. The tutor also begins the assessment process which relies heavily on ACTIN, the blackboard model of dynamic intent

inferencing (see Figure 5-4). The following three examples use the graphical blackboard display to illustrates the dynamics of the tutor in response to the student's actions during a real-time lesson scenario.

Missing Action. The support configuration for this lesson is shown in Figure 8-6.1. Two minutes before the acquisition of signal, the tutor posts the activity trees (i.e., function-subfunction-task structures) for configuration, verification and setup (Figure 8-6.2). The tutor hypothesizes that these activities should be the pre-pass activities the student attends to.

20 seconds later, the student performs an APSET action by selecting the "initialize" option in the control panel for the Application Processor object (Figure 8-6.3). The tutor receives this command and posts the APSET node on the blackboard and successfully connects the action to the XC_AP task. In other words, the tutor infers that that the APSET action is undertaken in support of the AP configuration task (Figure 8-6.4).

60 seconds later, the tutor first assesses the student's performance on the configuration function based on the blackboard configuration in Figure 8-6.4. The assessment is unsatisfactory because the student has not taken any action to support the TAC configuration tasks (XC_TAC). At this point, the tutor pauses the scenario, alerts the student with a beep, and turns the "ALERT" button to red.

In response to the tutor's alert, the student stops all activities and selects the "ALERT" button to find out what the problem is. The alert message specifies the unsatisfactory assessment as shown in Figure 8-6.5. The student next clicks on the "Explain" button which displays instructions for attending to the alert message. The student is instructed to select the "**Performance Assessments**" from the Status Information menu to display a list of information about the tutor's assessments (Figure 8-6.6). To learn more about the unsatisfactory assessment, the student selects "**Activity Assessments**" to display a panel of the same name (Figure 8-6.7). The panel shows that the CONFIG assessment done at time 17:30:26 is not "Okay". To rectify the problem, the student clicks on the "Explain" button to display details about the CONFIG assessment (Figure 8-6.8). According to the tutor, the action TACINIT is missing. If the student forgets what this action involves, the student can click on the line with the TACINIT word and then click "Explain" on the Assessment Details panel. A panel with the action sequence and description of TACINIT is shown (Figure 8-6.9) This panel is the same one used in the

Learning Operations by Doing lesson type. Either with or without help, the student proceeds to rectify the problem by performing the appropriate TACINIT action. When the student is ready to move on, the student selects "Continue" on the lesson panel. If there are no pending problems to be alerted, the tutor turns the "ALERT" button back to grey and restarts the scenario in real time. Otherwise, the tutor alerts the student again with a beep, and the process of attending the alert repeats.

Repeated Action. Another situation that the tutor responds to is illustrated in Figure 8-6.10. The blackboard shows that the NCCHEK action has been posted twice. That is, the student unnecessarily repeated the request for operational data messages (ODMs) from the Network Control Center. When the tutor assesses the VERIFY function, the repeated action is detected and the unsatisfactory assessment alerted. The student sees the problem in the assessment panels as shown in Figure 8-6.11. In this case, the problem cannot be rectified. As long as the student attends to the alert message (by learning about the problem), the student can proceed by selecting "Continue".

Uninterpreted Action. Figure 8-6.12 shows a situation in which a student action is posted but not connected to any activity trees on the blackboard. Specifically, the tutor is unable to infer the student's intent to display the spacecraft's Power Subsystem panel at this time. The tutor issues an alert message as shown in Figure 8-6.13. The information on the uninterpreted action is shown in Figure 8-6.14.

The tutor offers three other features to foster the student's understanding of the tutor, and to help the student manage the many operator activities at hand. First, the blackboard display used so far in the above examples is also available to the student through the "**Action Interpreter (ACTIN)**" option on the Tutor's Assessments panel (e.g., Figure 8-6.14). The blackboard is also inspectable. The student can click on any node to obtain more information (Figure 8-6.15). Second, the tutor's diagnostic actions and blackboard activities are recorded in a log that is accessible through the option "**Tutorial Events**" of the "Status Information" pulldown menu on the lesson panel (Figure 8-6.16). Third, the tutor records all student actions in a log that is accessible through the option "**Student Actions**" of the same pulldown menu (Figure 8-6.17).

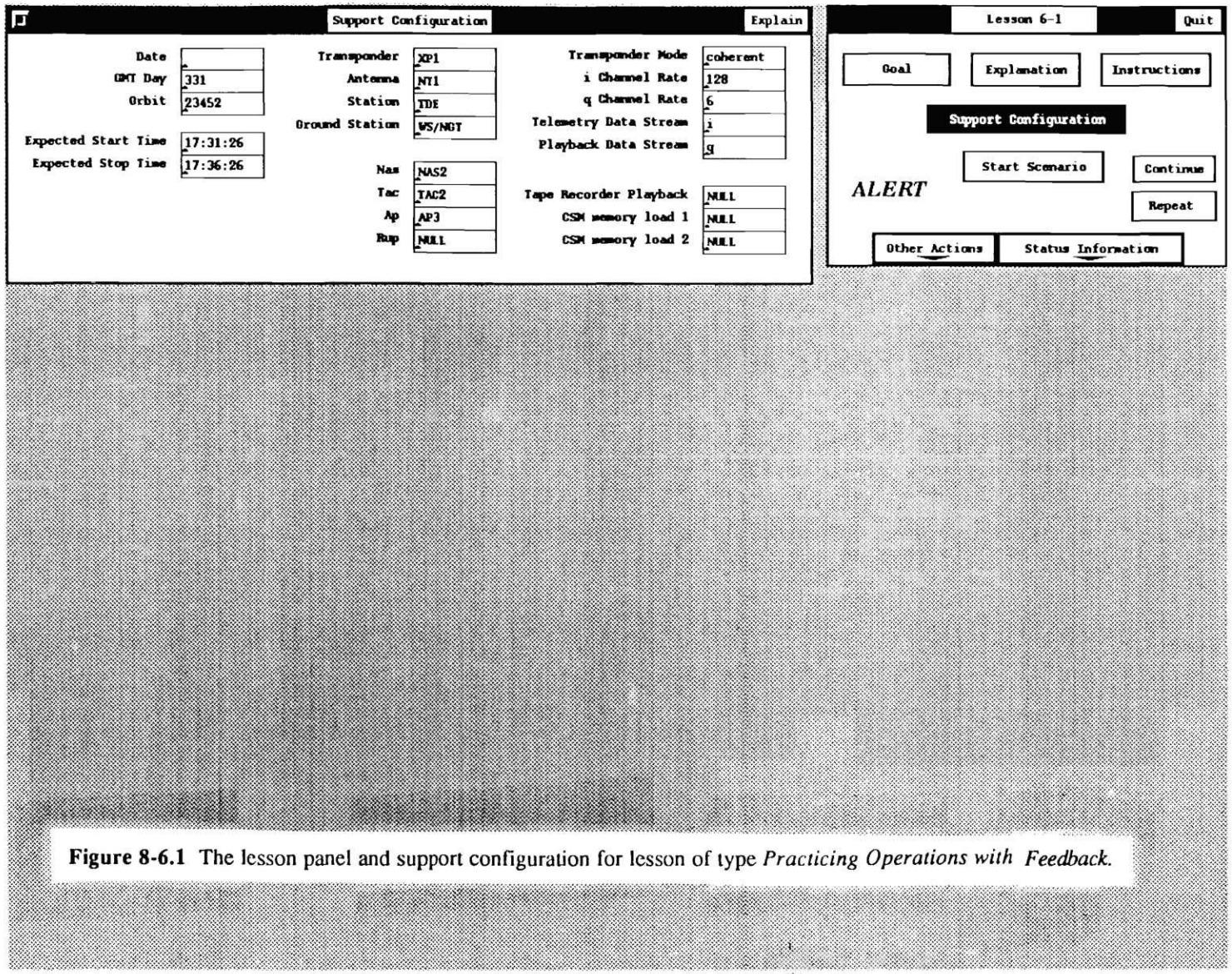


Figure 8-6.1 The lesson panel and support configuration for lesson of type *Practicing Operations with Feedback.*

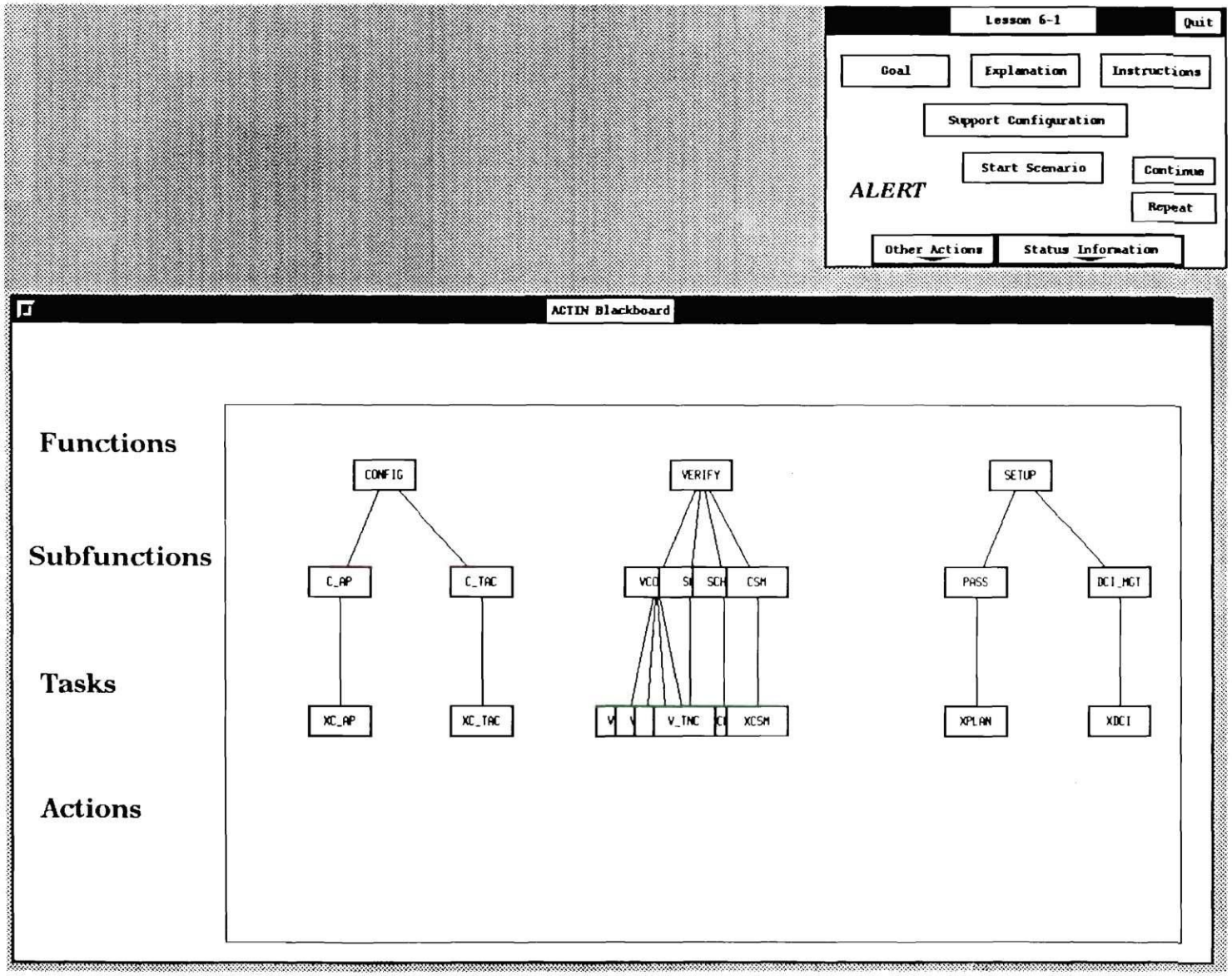


Figure 8-6.2 The blackboard before the start of a support for lesson of type *Practicing Operations with Feedback.*

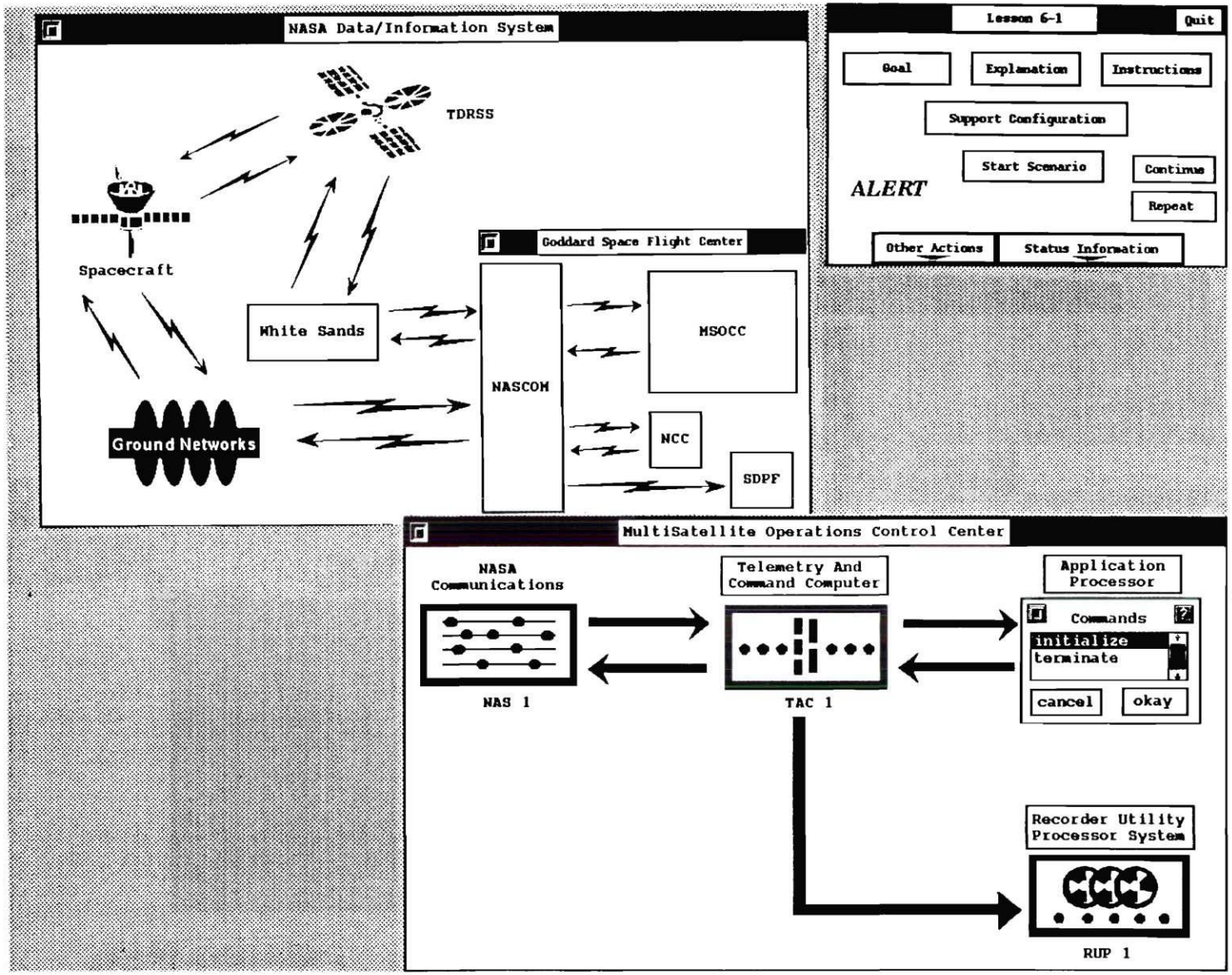


Figure 8-6.3 The student's APSET action for lesson of type *Practicing Operations with Feedback*.

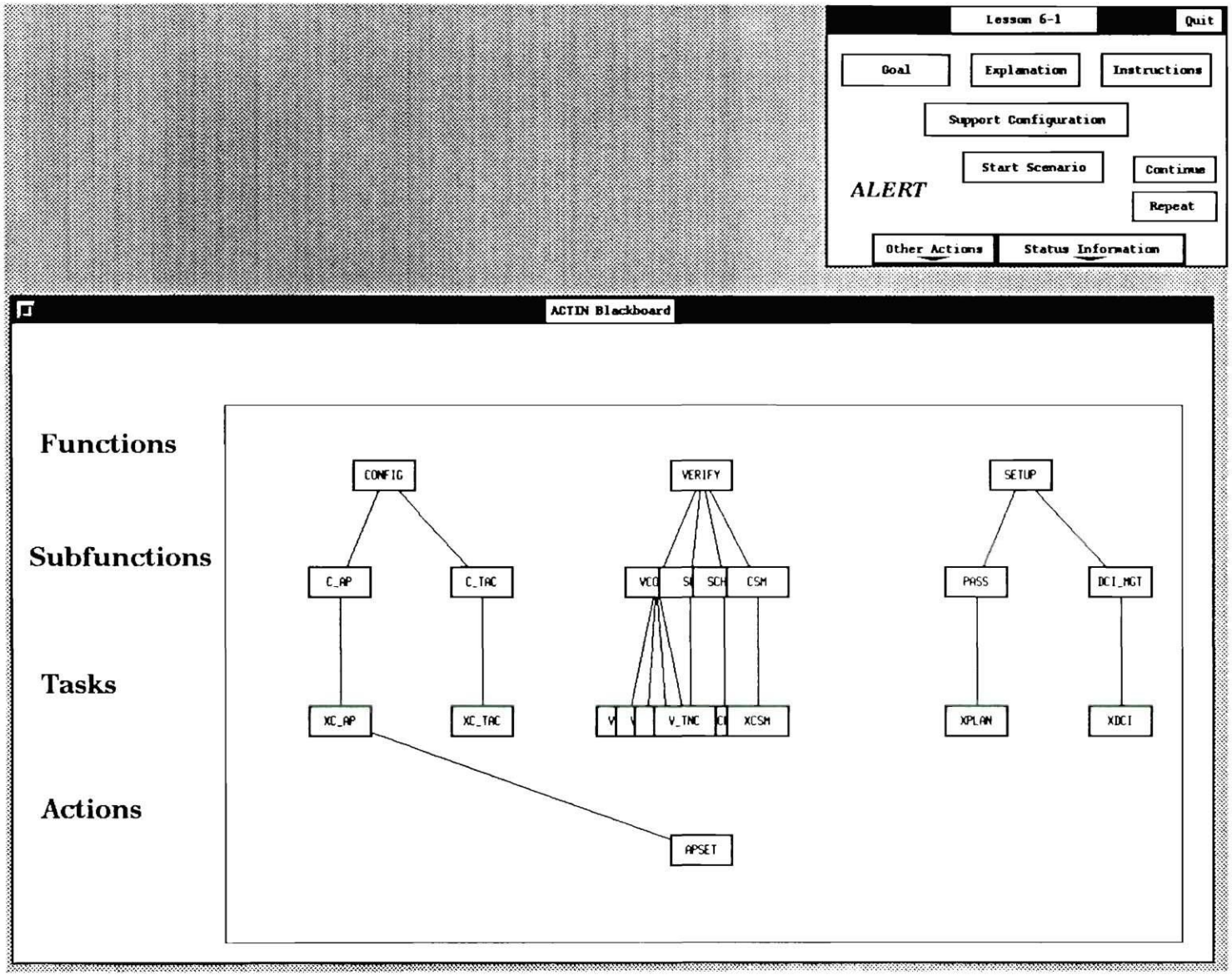


Figure 8-6.4 The blackboard with the APSET action for lesson of type *Practicing Operations with Feedback.*

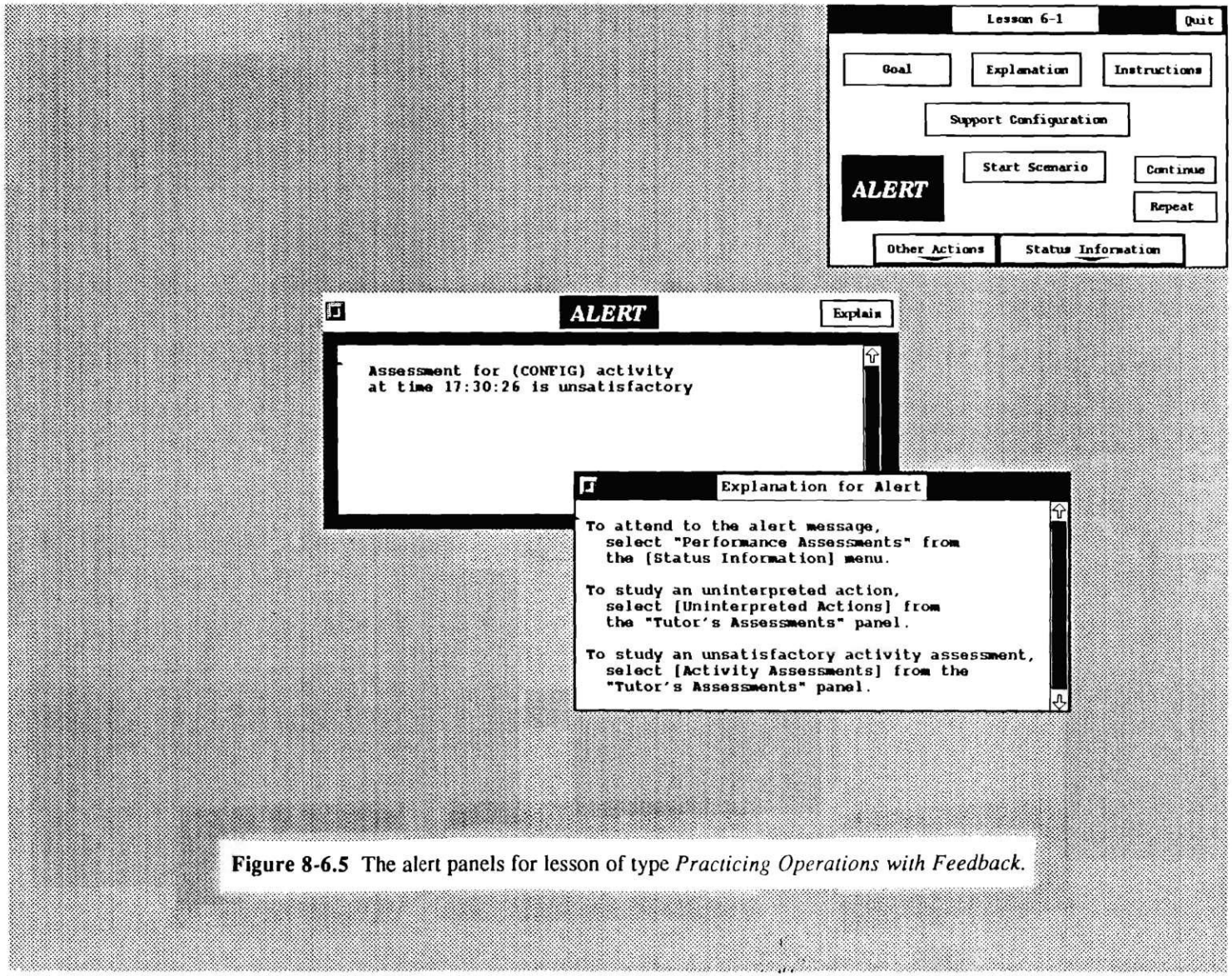


Figure 8-6.5 The alert panels for lesson of type *Practicing Operations with Feedback.*

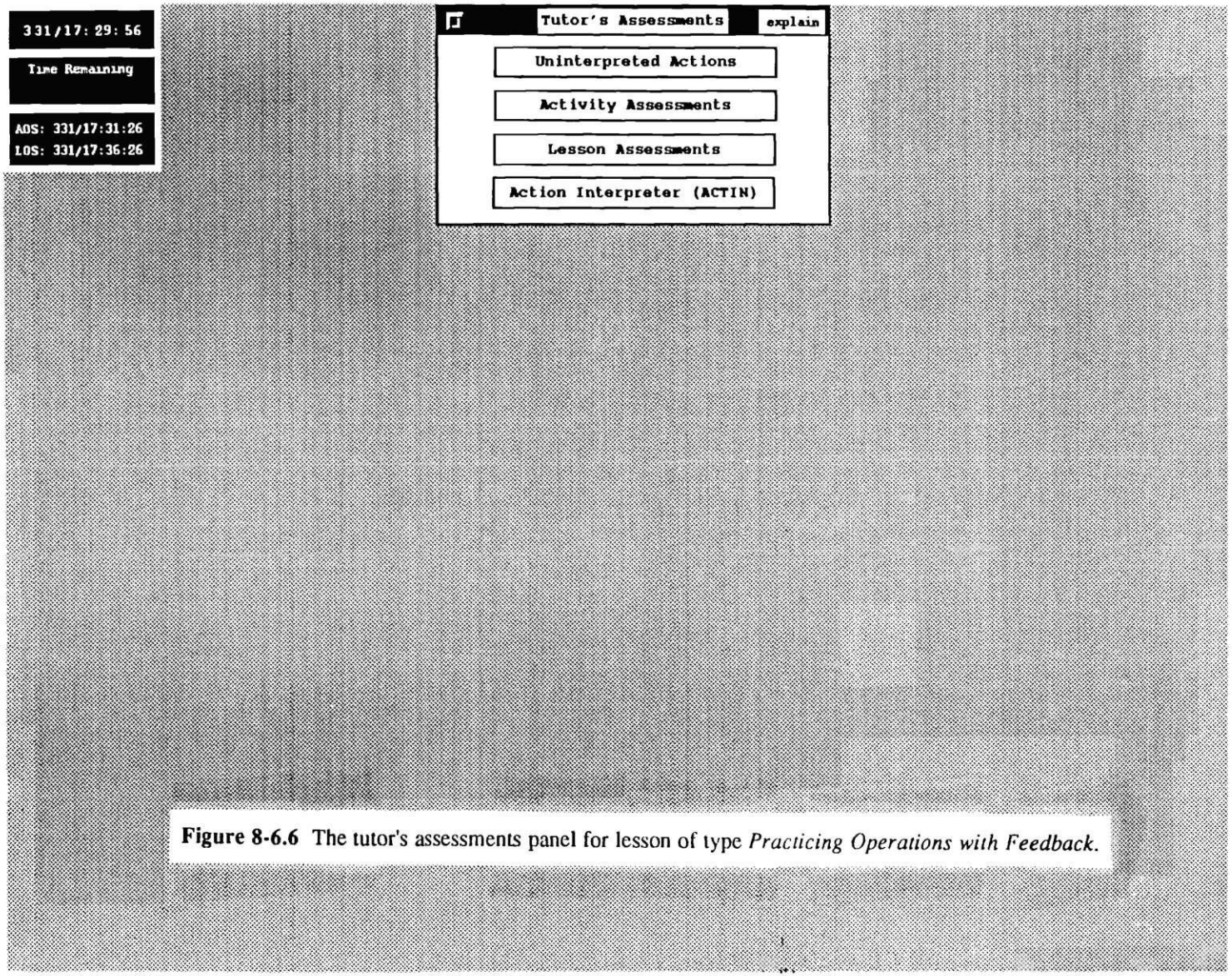


Figure 8-6.6 The tutor's assessments panel for lesson of type *Practicing Operations with Feedback.*

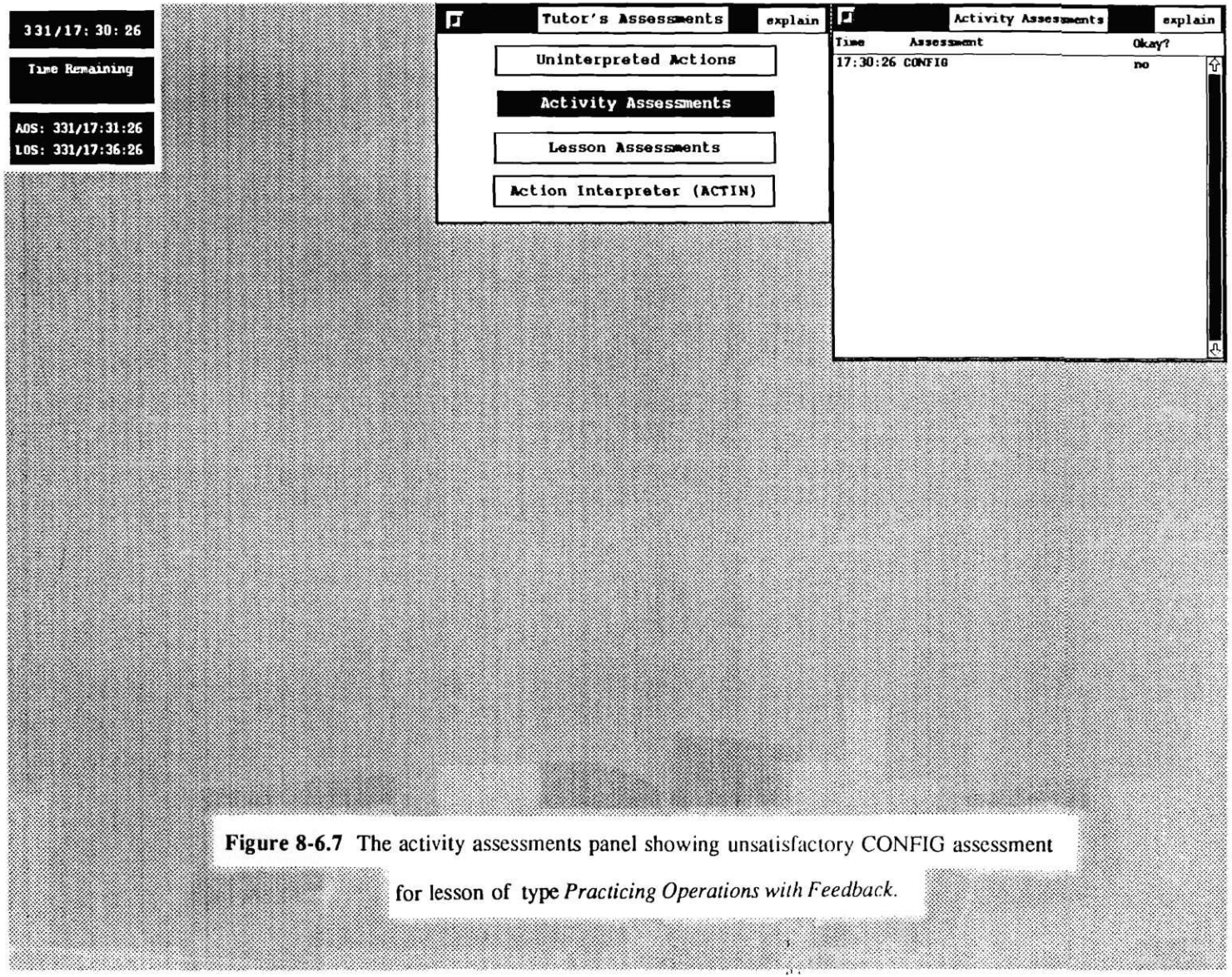


Figure 8-6.7 The activity assessments panel showing unsatisfactory CONFIG assessment for lesson of type *Practicing Operations with Feedback.*

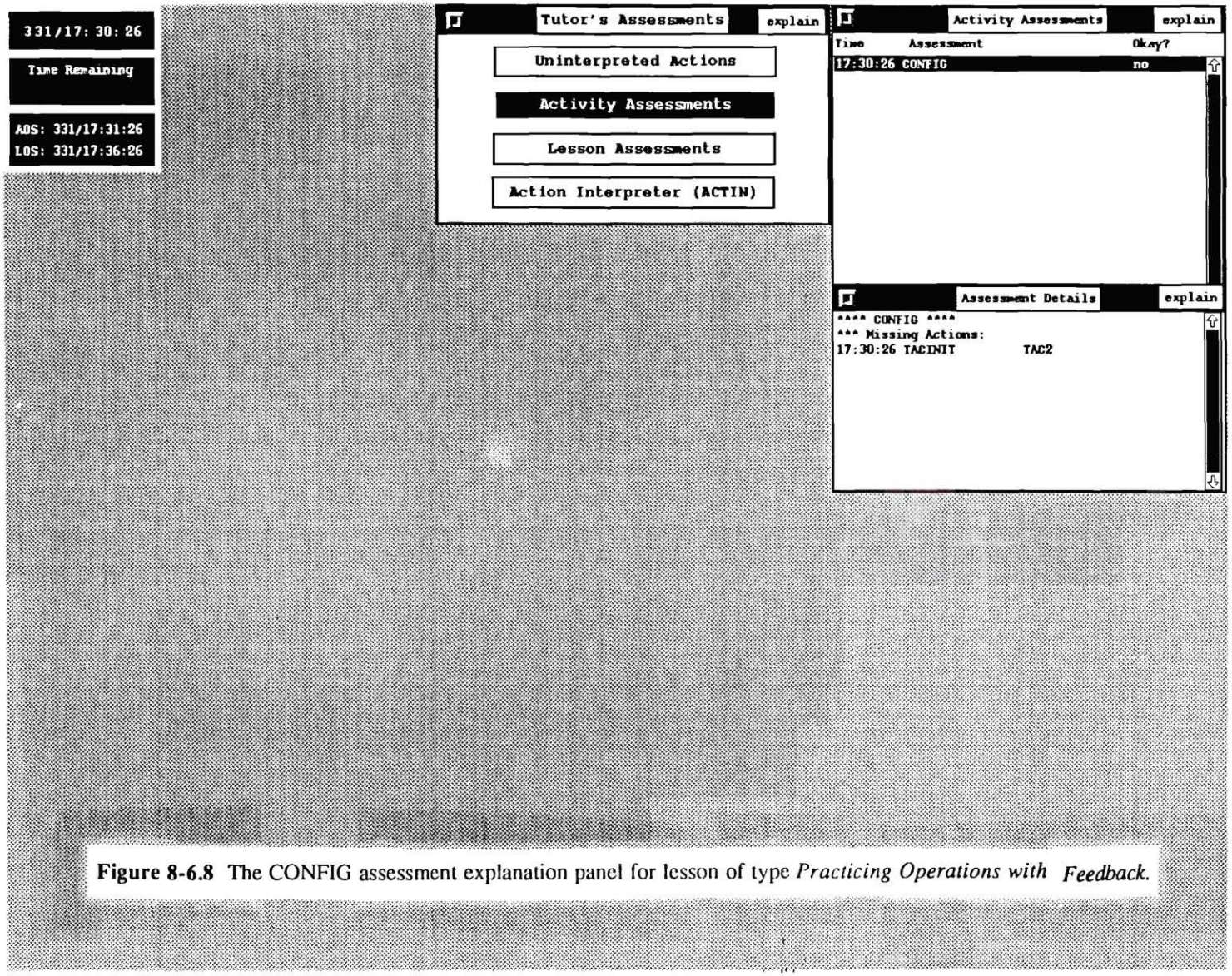


Figure 8-6.8 The CONFIG assessment explanation panel for lesson of type *Practicing Operations with Feedback.*

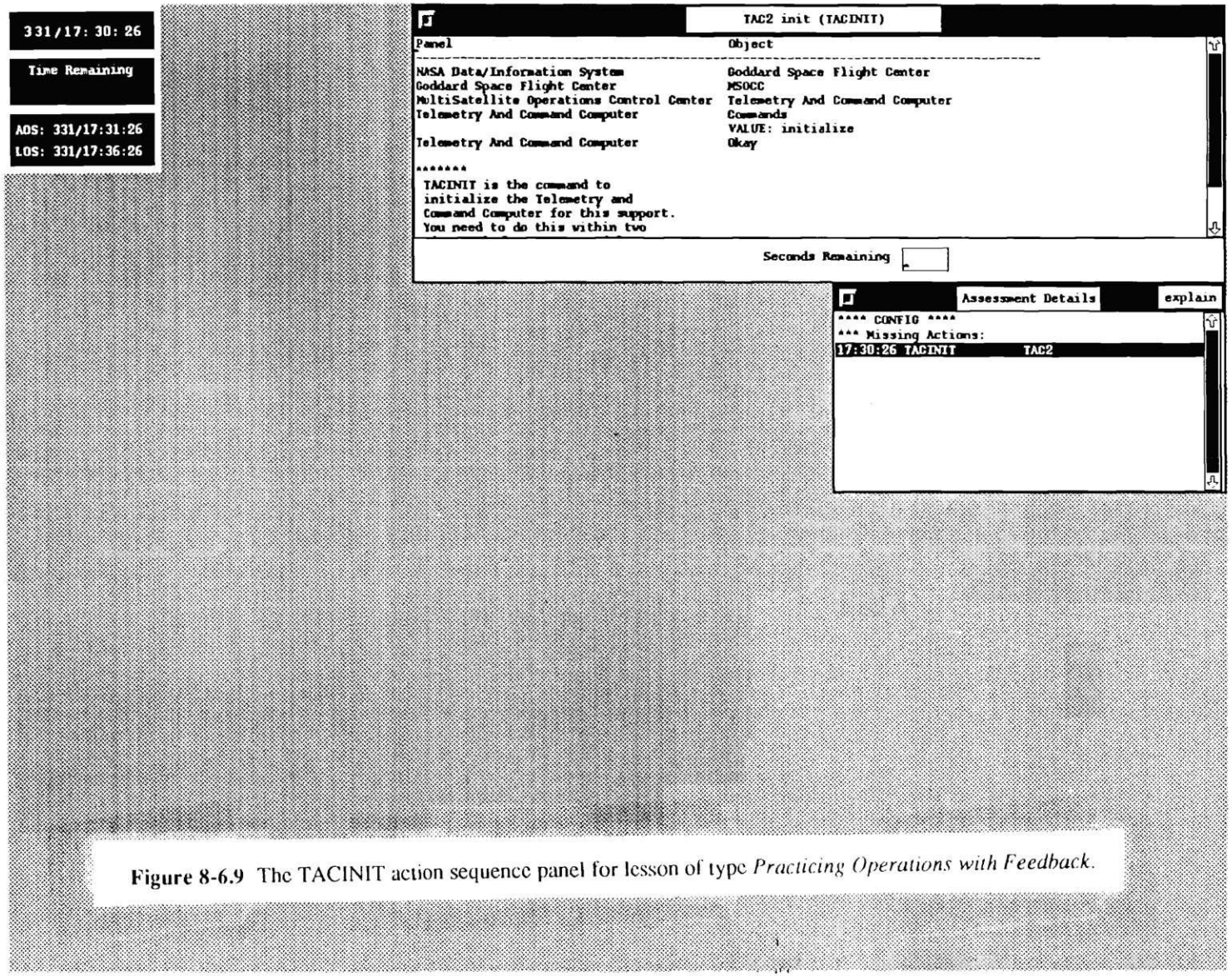


Figure 8-6.9 The TACINIT action sequence panel for lesson of type *Practicing Operations with Feedback.*

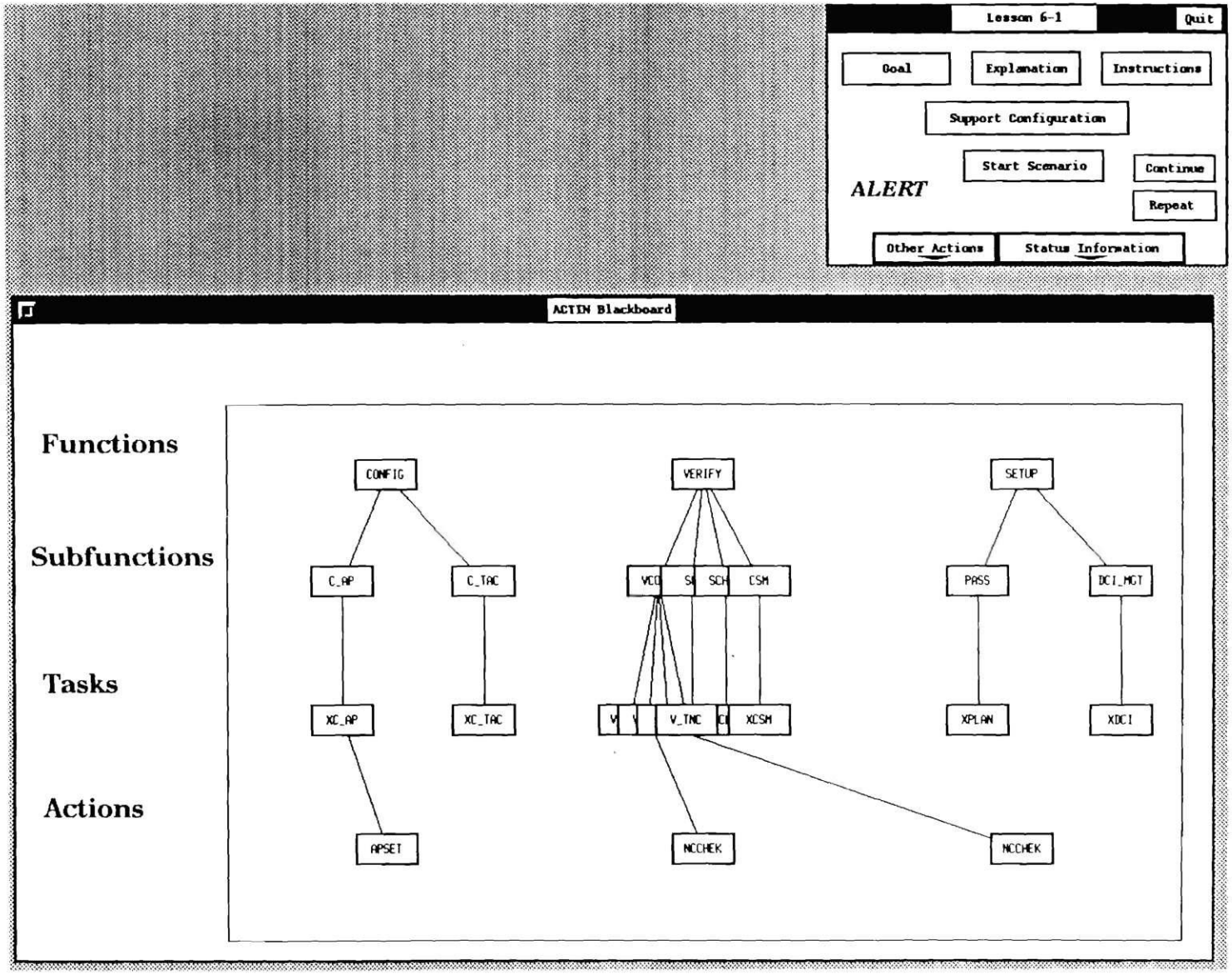


Figure 8-6.10 The blackboard with repeated NCCHEK actions for lesson of type *Practicing Operations with Feedback.*

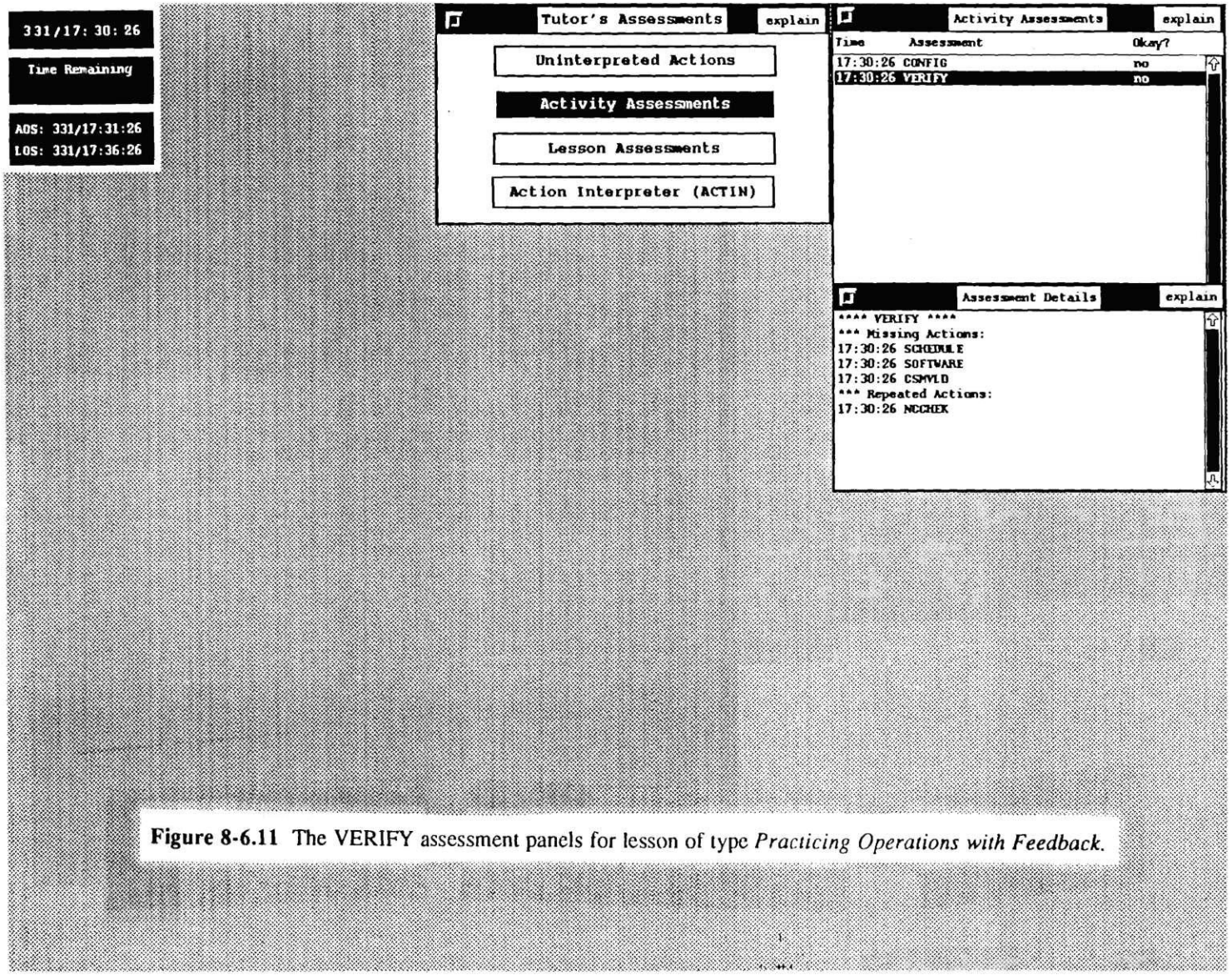


Figure 8-6.11 The VERIFY assessment panels for lesson of type *Practicing Operations with Feedback.*

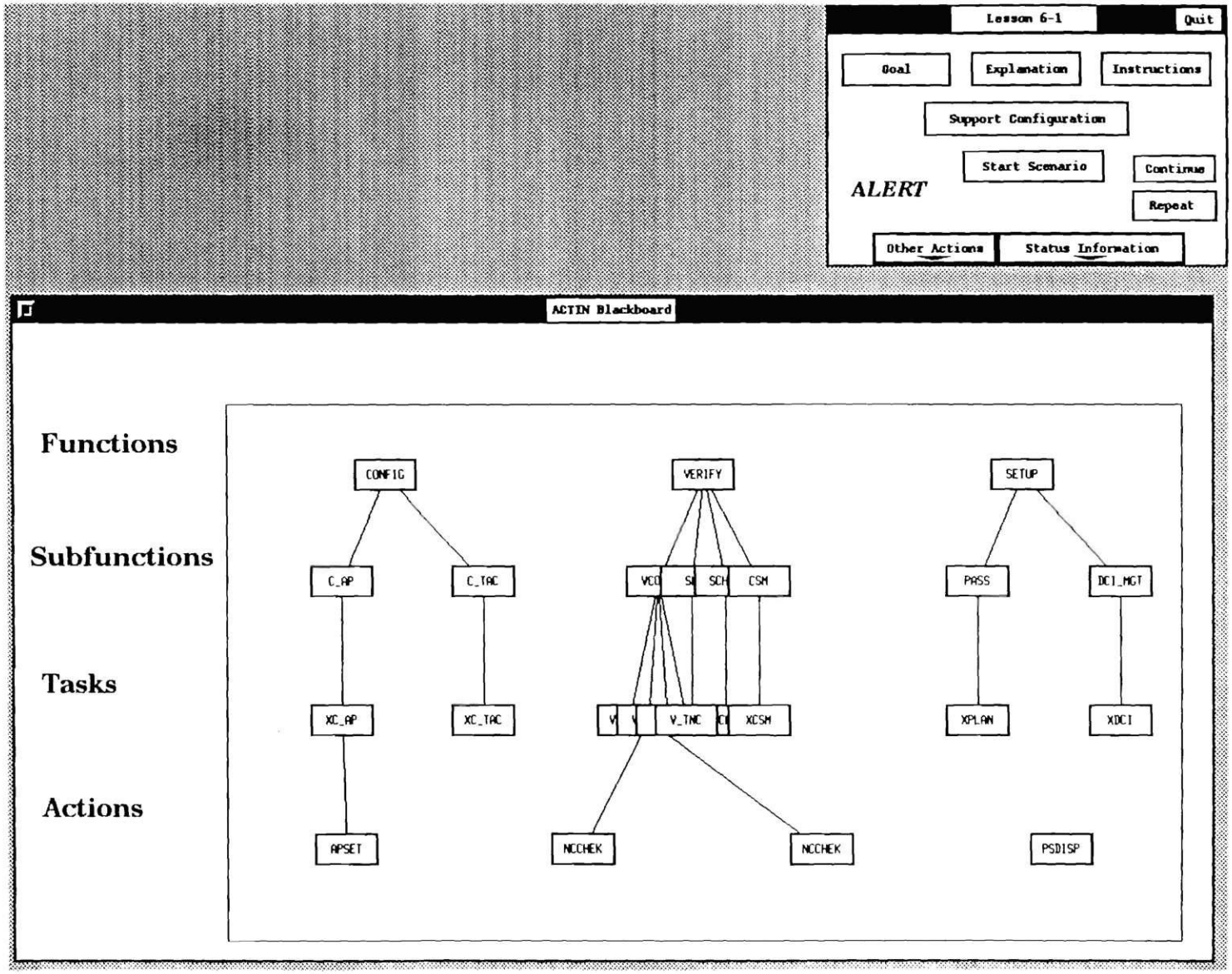


Figure 8-6.12 The blackboard with uninterpreted PSDISP action for lesson of type *Practicing Operations with Feedback.*

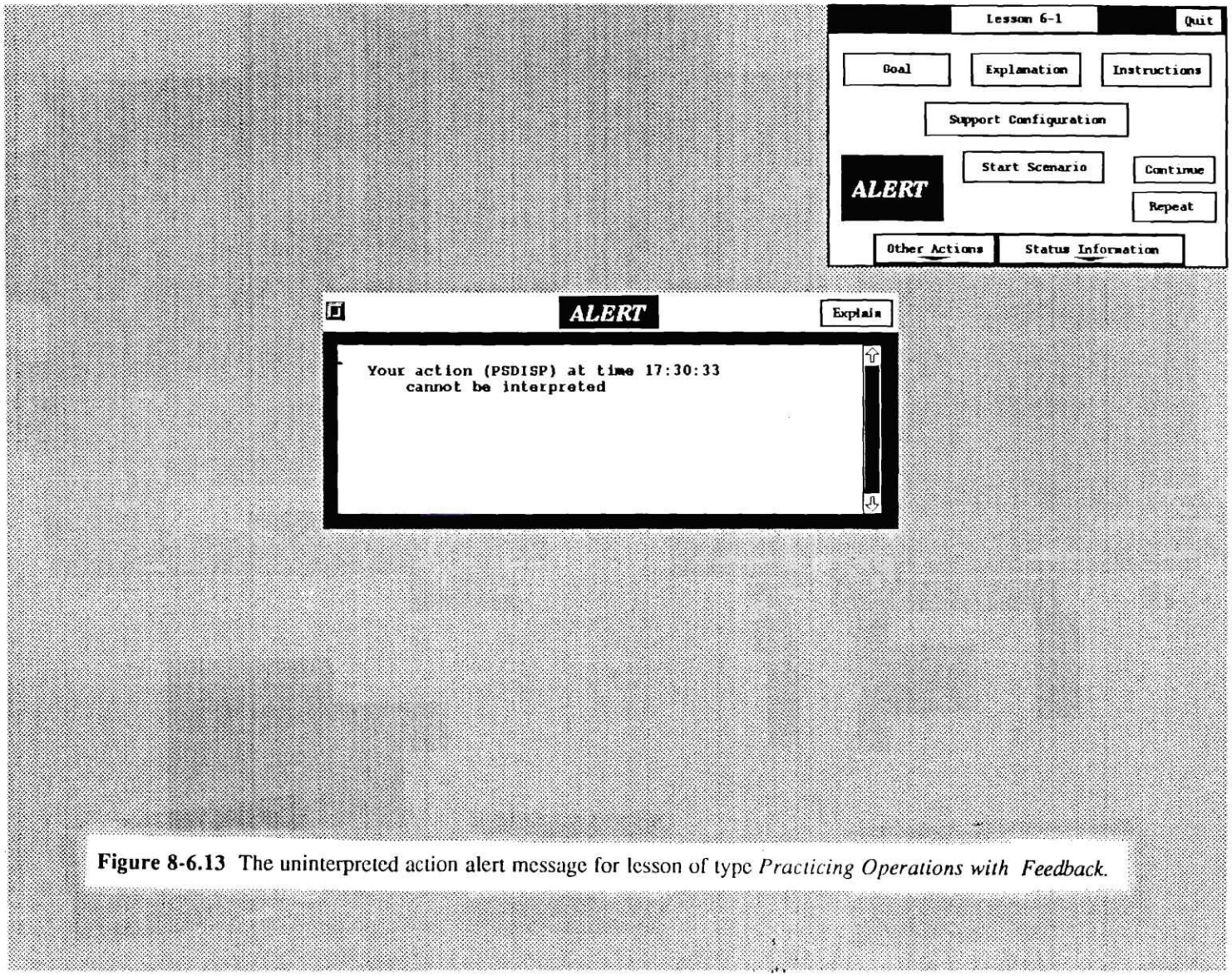


Figure 8-6.13 The uninterpreted action alert message for lesson of type *Practicing Operations with Feedback.*

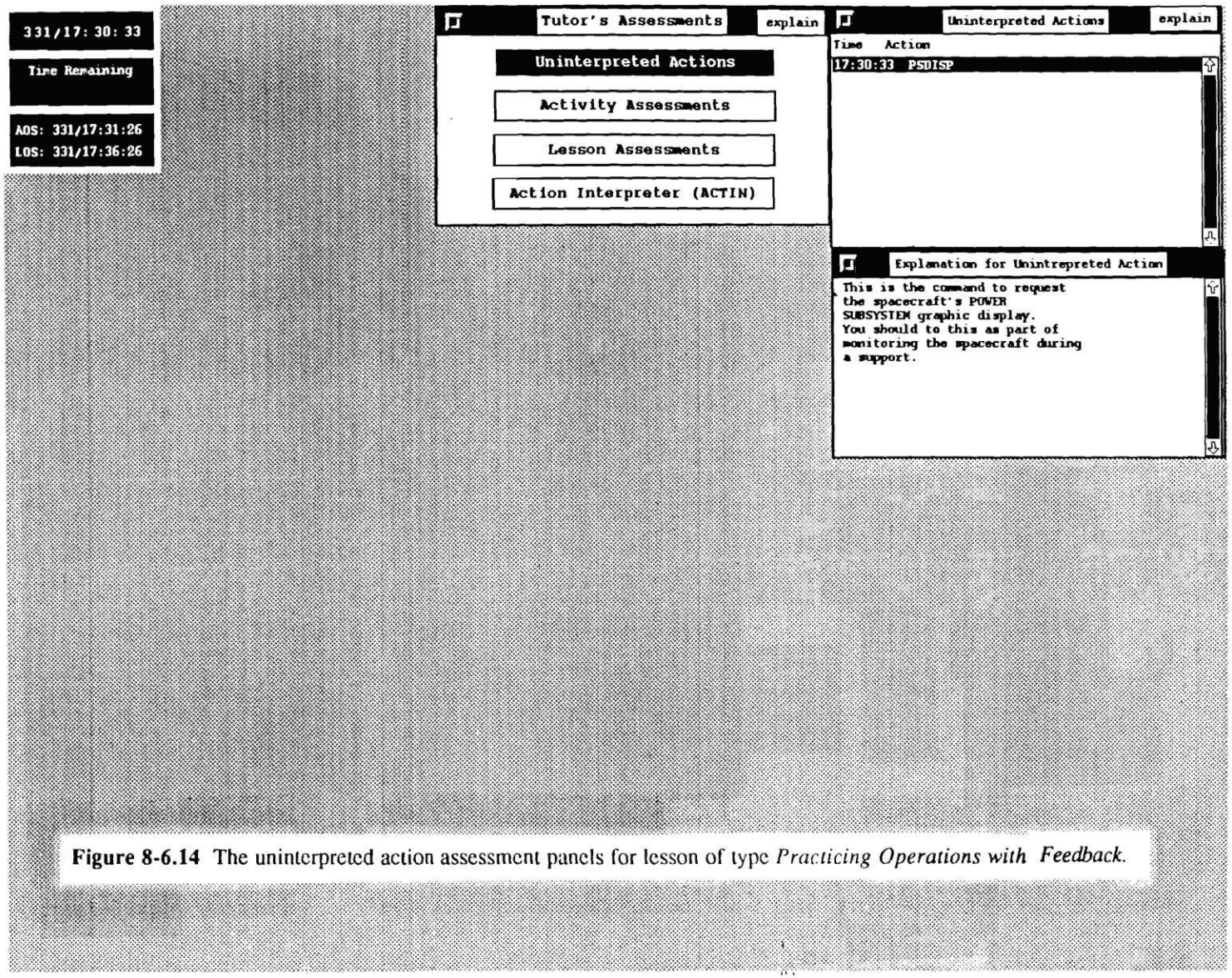


Figure 8-6.14 The uninterpreted action assessment panels for lesson of type *Practicing Operations with Feedback.*

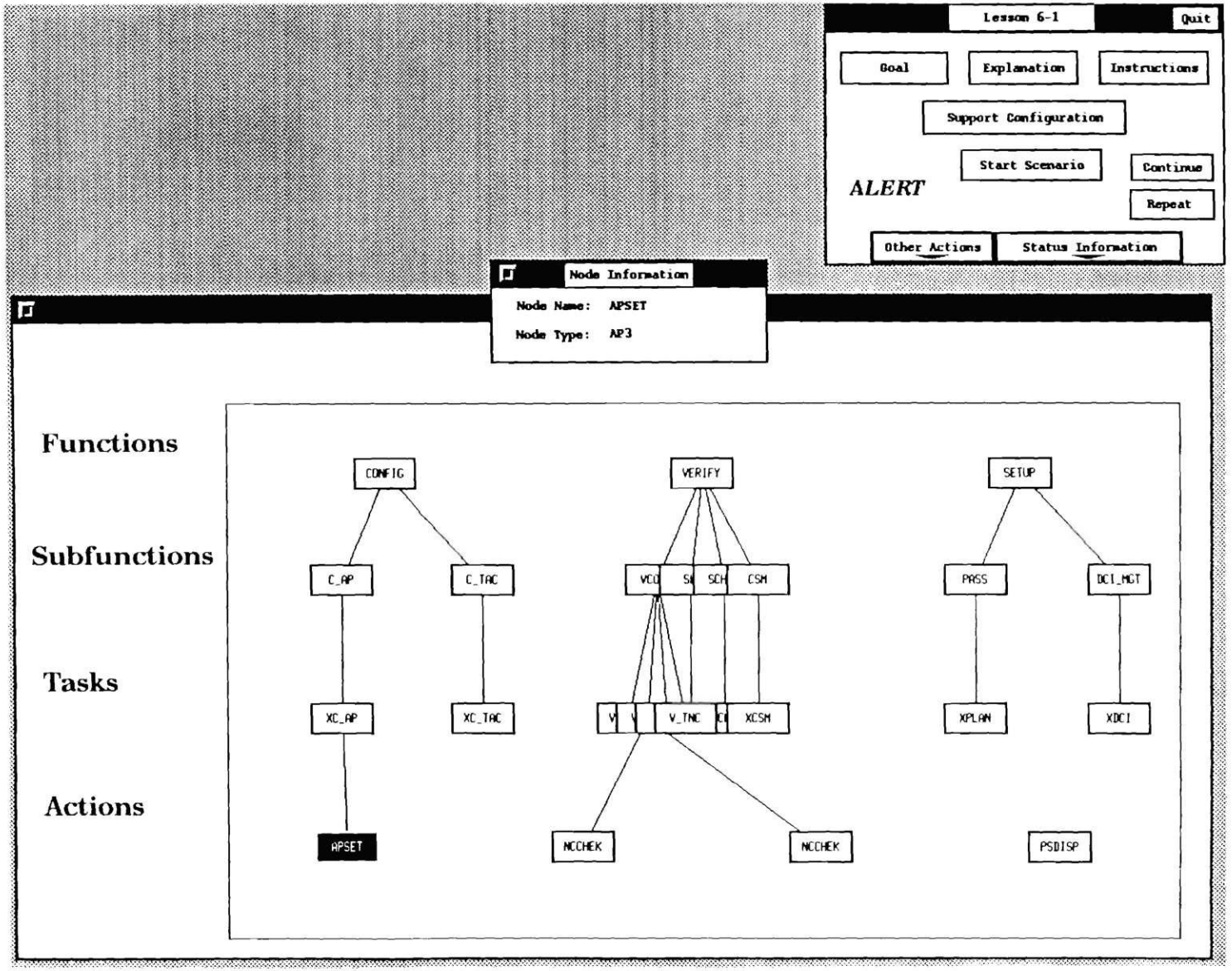


Figure 8-6.15 The blackboard with highlighted node and node information for lesson of type *Practicing Operations with Feedback.*

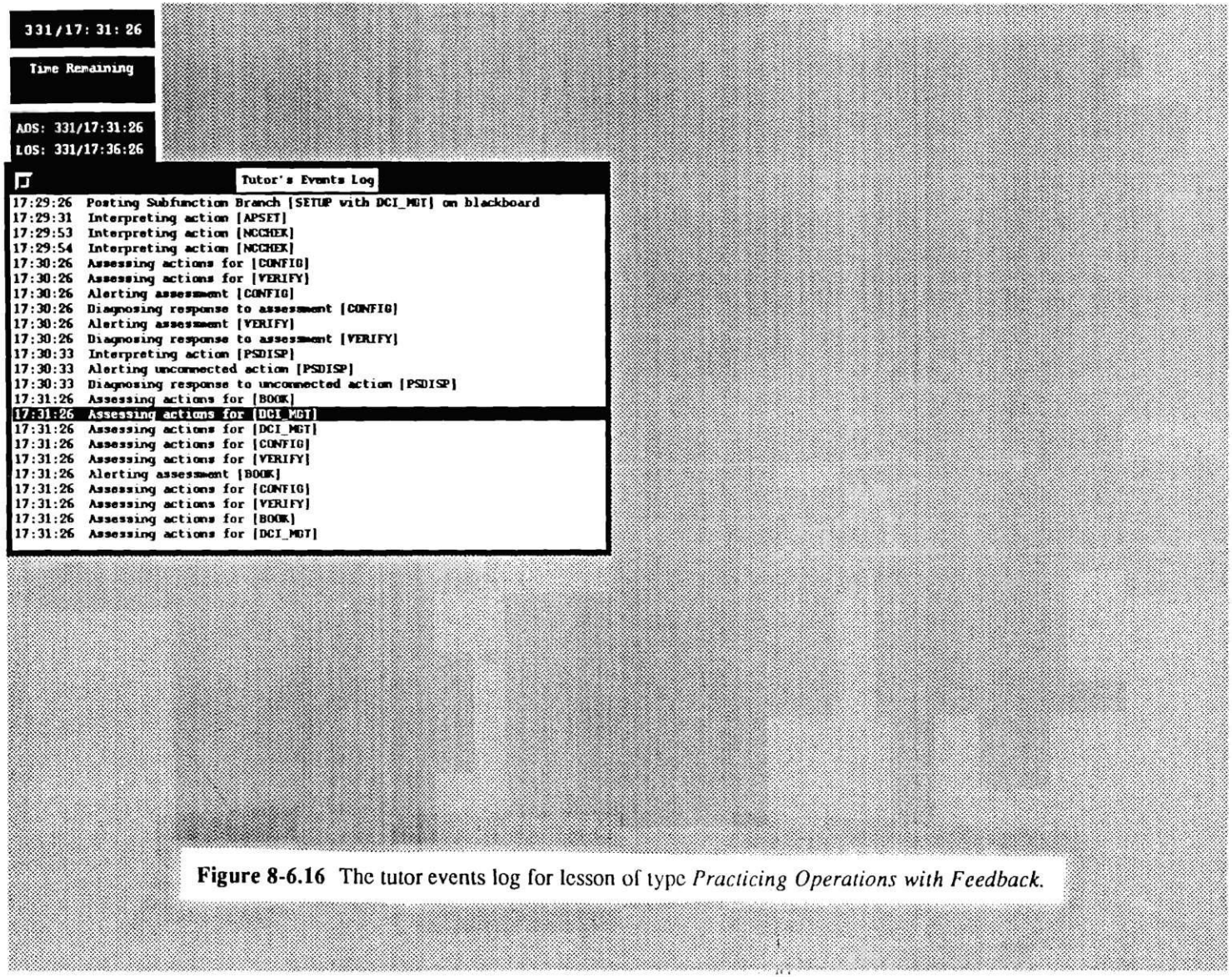


Figure 8-6.16 The tutor events log for lesson of type *Practicing Operations with Feedback.*

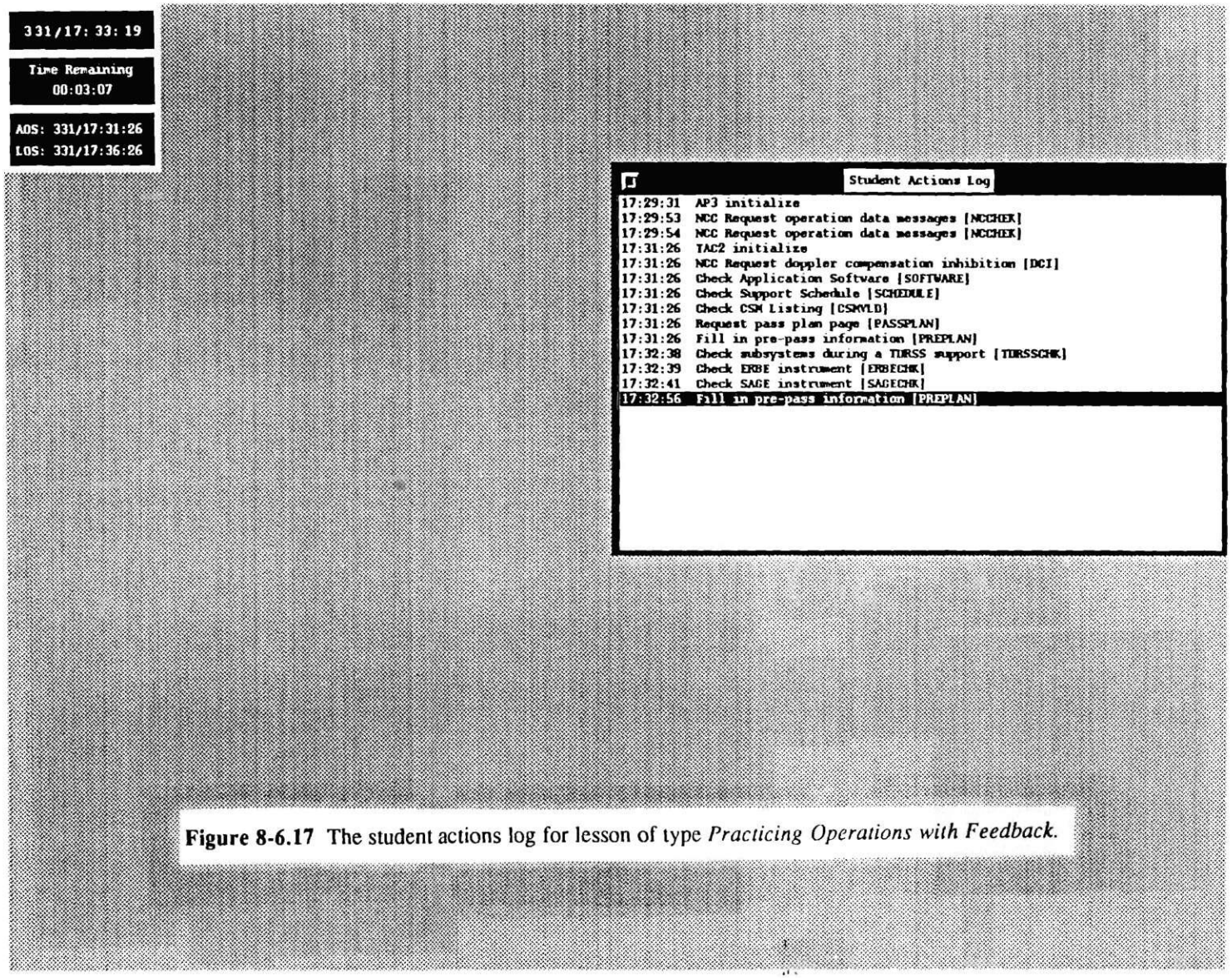


Figure 8-6.17 The student actions log for lesson of type *Practicing Operations with Feedback.*

Besides providing immediate feedback, the tutor also ensures that the support scenario is proceeding properly. The tutor does so by performing overall blackboard assessments for pre-pass, on-pass, and post-pass activities at three checkpoints: right before the pass begins, right after the pass ends, and shortly after the pass is over respectively. This feature of the tutor is more relevant in the next lesson type and will be discussed further then.

The lesson ends when the scenario ends. If the student decides to repeat the lesson, a different support is configured so the student can practice the FOT operator activities again. From the student's point of view, the ultimate goal is to manage an entire support with minimal feedback from the tutor.

Lesson Type 7: Practicing Operations with Checkpoints

Just as in the previous lesson, the goal is for the student to perform all FOT activities in a real-time, practice environment. The difference here is that the tutor only intervenes at the three pre-pass, on-pass and post-pass checkpoints. In this way, the student has a chance to recover from the tutor's unsatisfactory assessments without the tutor's interventions.

Example

Figure 8-7.1 shows the lesson panel for this lesson type. This lesson panel does not have the "ALERT" button. The dynamics of this lesson can be illustrated with the first example used previously, focusing on the CONFIG function. After the first unsatisfactory assessment about the CONFIG function (Figure 8-6.4), instead of alerting the student, the tutor just posts the result on the Activity Assessments panel as before (Figure 8-6.7). The student has the same access to all assessment panels, except now it is optional.

A few seconds later, the student correctly performs the TACINIT action which is posted and connected to the TAC configuration task (Figure 8-7.2). Right before the start of the pass, the tutor reassesses all pre-pass activities. At this time, the CONFIG function has been properly completed but not the other pre-passfunctions as shown in Figure 8-7.3. The tutor alerts the student with a message shown in

Figure 8-7.4, and makes a button labeled "**Pending Actions**" visible on the lesson panel. When the student clicks on this button, a list of student actions that have not been taken is shown (Figure 8-7.5). The student can get help for these actions the same way as in previous lesson by inspecting the assessment details (e.g., Figure 8-6.9). In any case, the student is required to perform all pending actions before the support will proceed. As each action is performed, it is removed from the Pending Actions panel. When the list becomes empty, the button goes away and the scenario resumes in real time.

During the course of the support, the student is encouraged to inspect the tutor's assessments to enhance self-monitoring and self-recovery skills. Figure 8-7.6 shows a series of activity assessments performed by the tutor.

This lesson also offers the same set of features to view the blackboard, a history of tutor's actions and a history of the student's actions. This lesson, when repeated, enables the student to practice all FOT operations in a different support scenario. From the student's viewpoint, the ultimate goal is to manage an entire support without any interventions from the tutor.

This concludes the discussion on the student-tutor interactions of GT-VITA in terms of the seven lesson types. The GT-VITA design architecture and implementation details, and a discussion on GT-VITA with respect to the tutor/aid paradigm are presented in the next chapter.

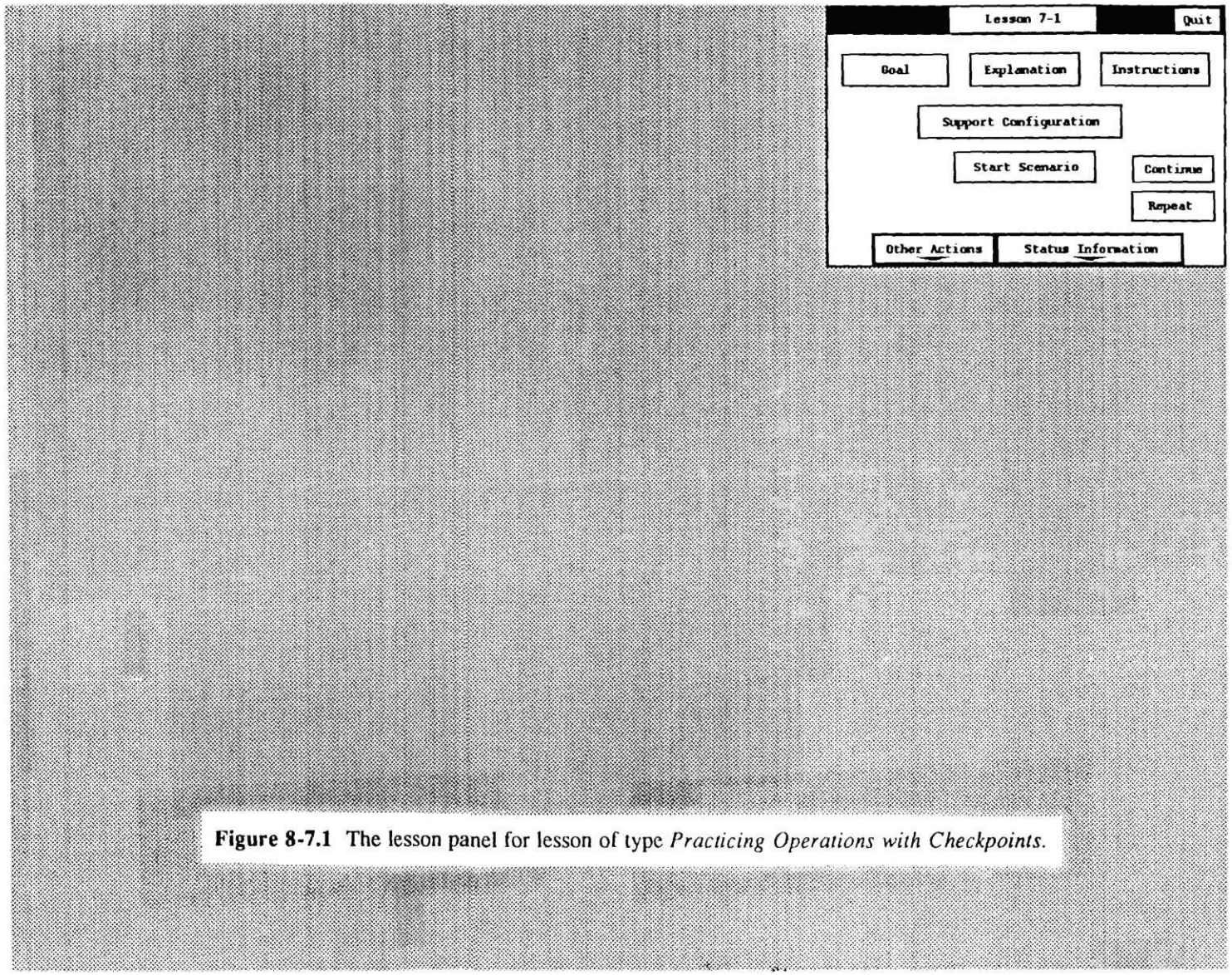


Figure 8-7.1 The lesson panel for lesson of type *Practicing Operations with Checkpoints.*

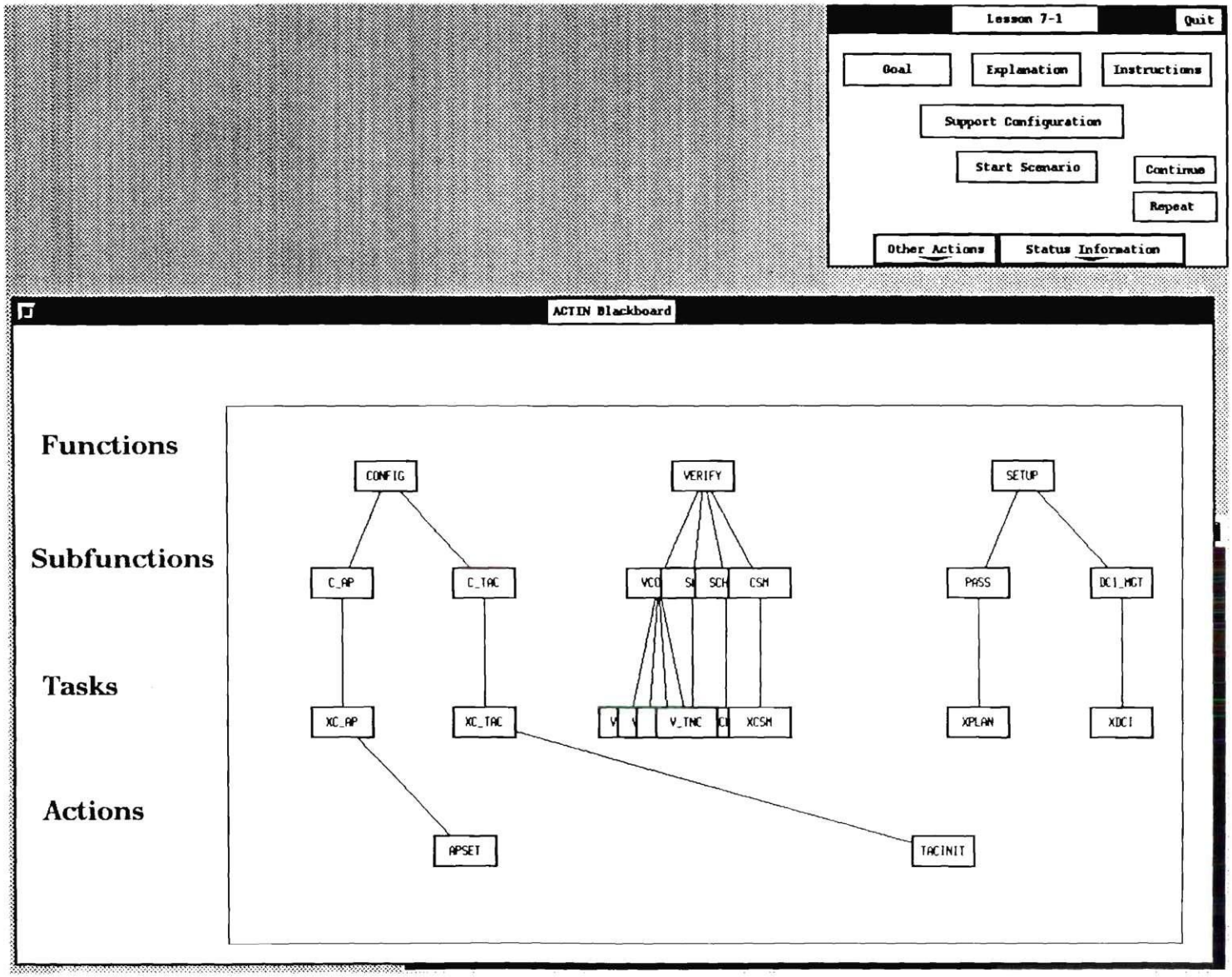


Figure 8-7.2 The blackboard with the TACINIT action for lesson of type *Practicing Operations with Checkpoints.*

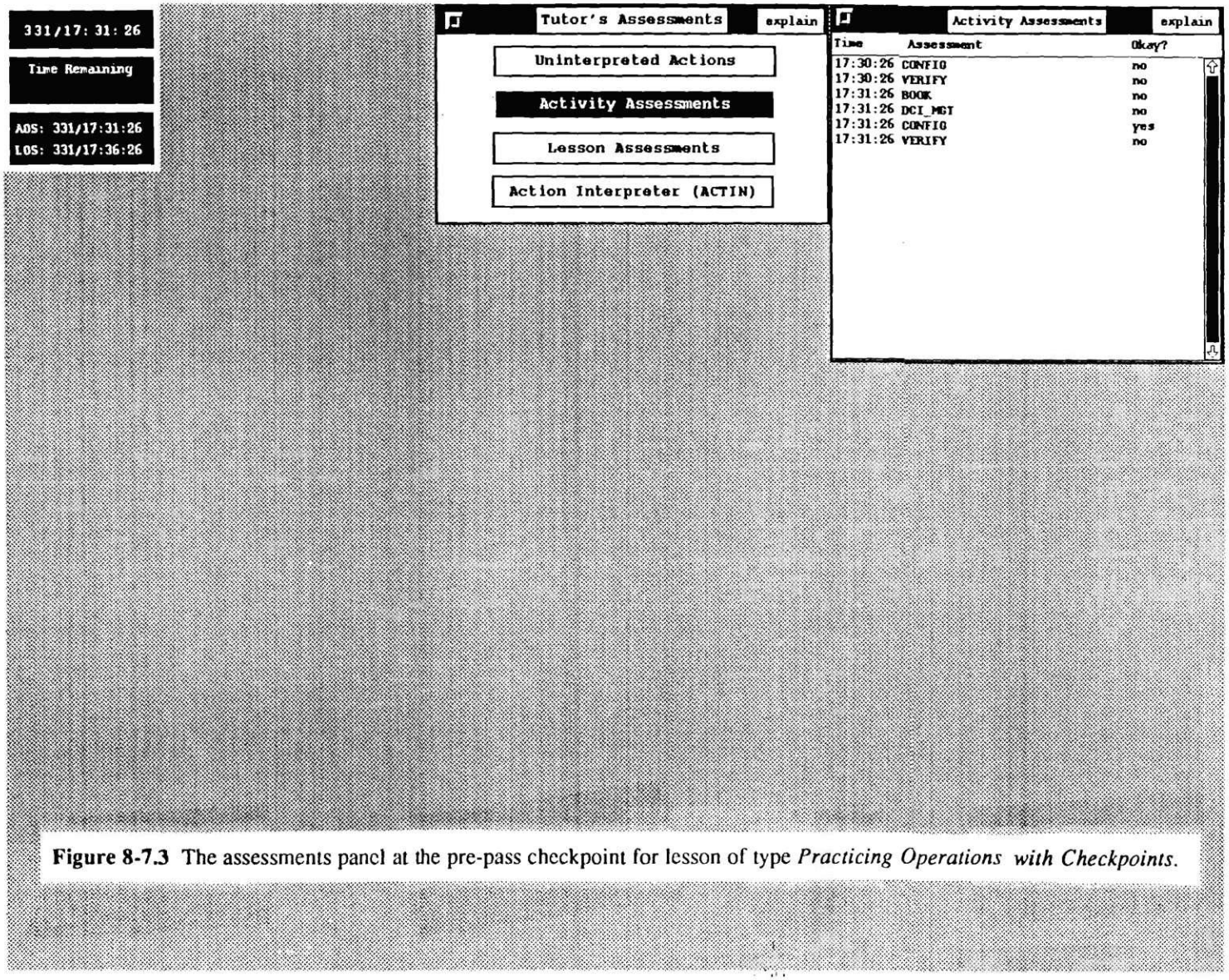


Figure 8-7.3 The assessments panel at the pre-pass checkpoint for lesson of type *Practicing Operations with Checkpoints.*

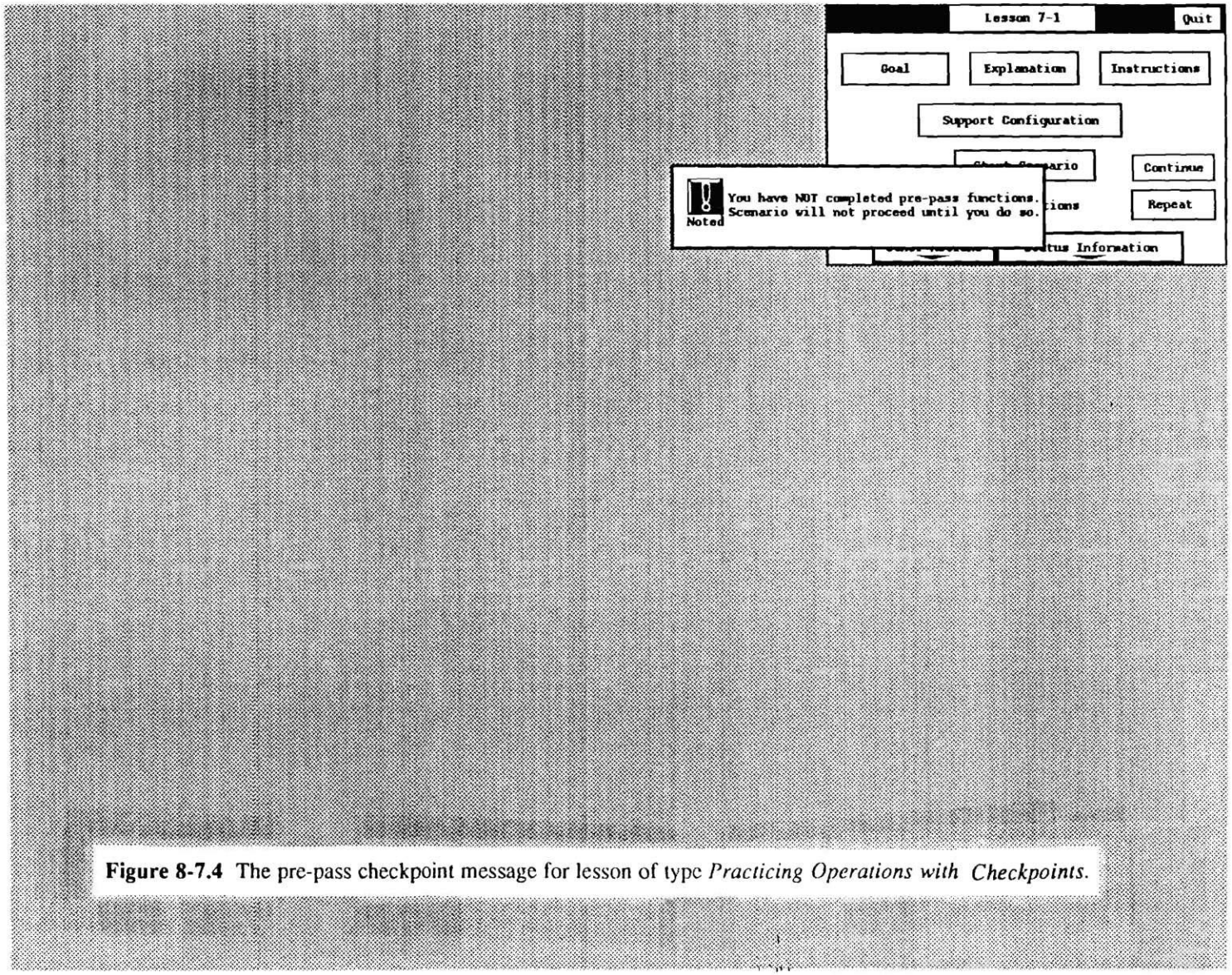


Figure 8-7.4 The pre-pass checkpoint message for lesson of type *Practicing Operations with Checkpoints.*

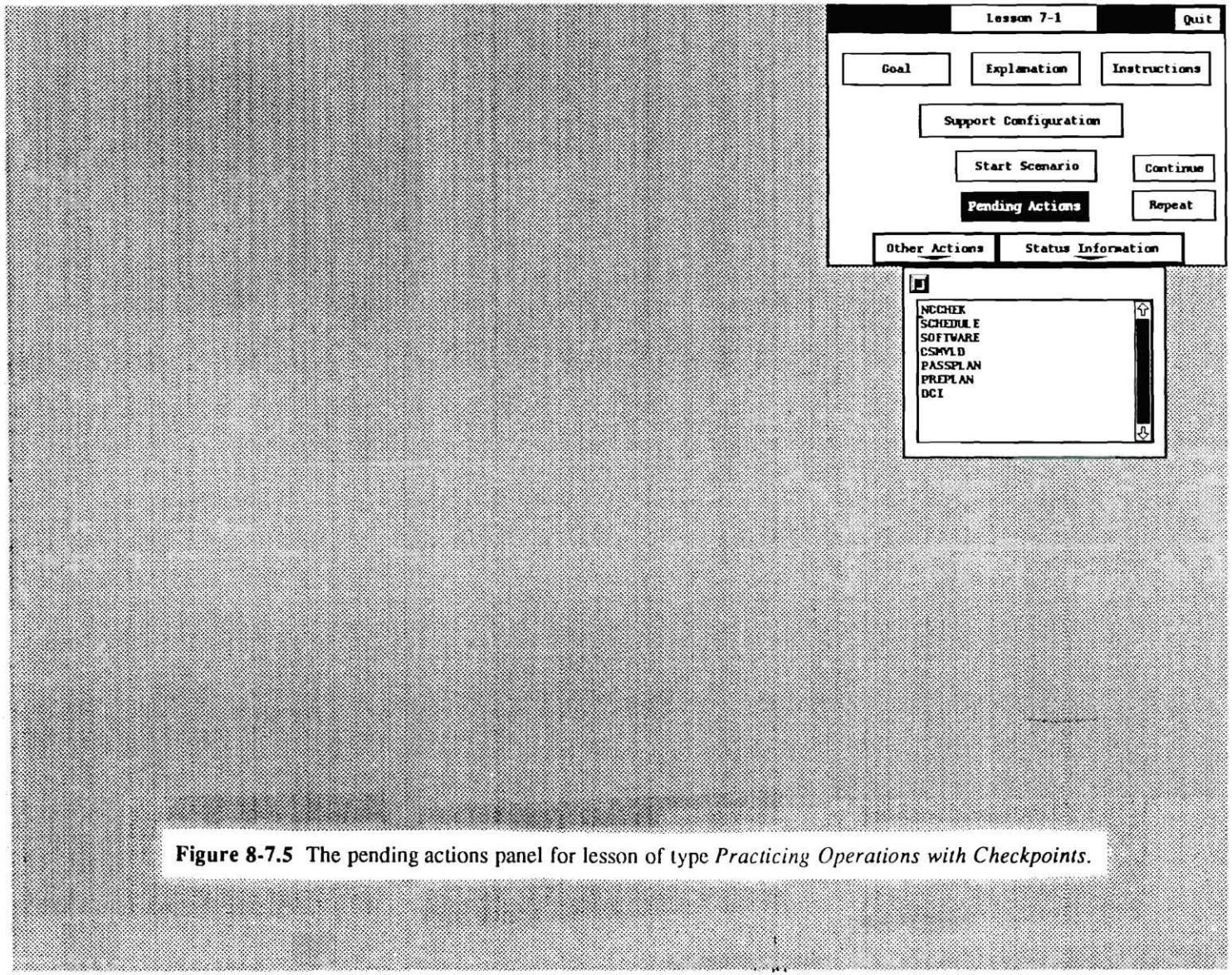


Figure 8-7.5 The pending actions panel for lesson of type *Practicing Operations with Checkpoints.*

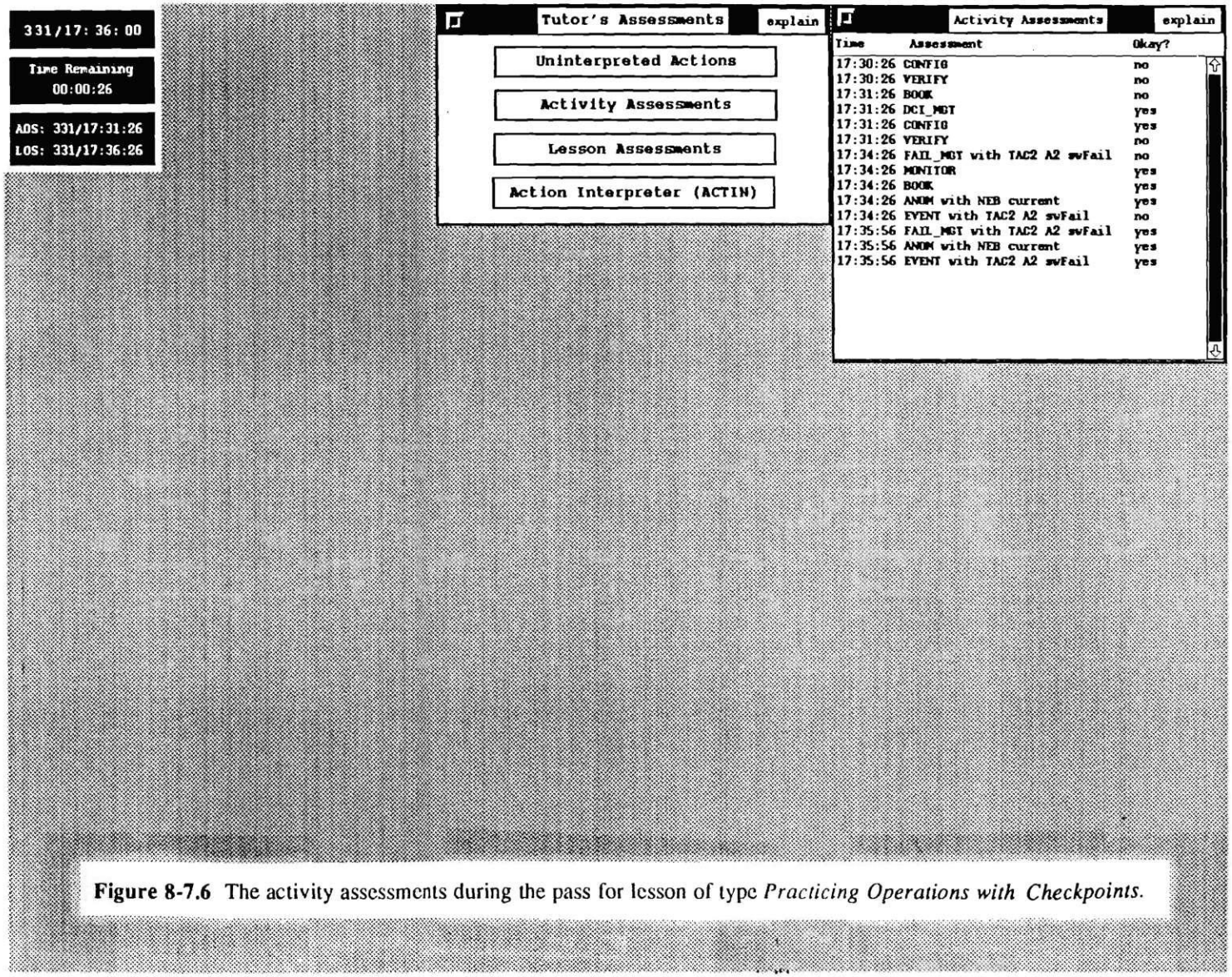


Figure 8-7.6 The activity assessments during the pass for lesson of type *Practicing Operations with Checkpoints.*

CHAPTER IX

IMPLEMENTATION OF GT-VITA, THE GEORGIA TECH VISUAL INSPECTABLE TUTOR AND ASSISTANT FOR THE GEORGIA TECH PAYLOAD OPERATIONS CONTROL CENTER

PART III: DESIGN AND DISCUSSION

Chapter VII describes the user interfaces of GT-VITA and Chapter VIII details the student-tutor interactions of GT-VITA. In this chapter, how GT-VITA works is further described from a design standpoint instead of from the student's standpoint. This chapter concludes the three-part discussion of the implementation of GT-VITA.

GT-VITA Implementation Architecture

GT-VITA is implemented in C++ as a separate process from the GT-POCC simulation and the GT-POCC task interface. The graphical, interactive user interfaces of GT-VITA are developed with TAE Plus (1990). GT-VITA is developed with the enhanced OFMspert architecture discussed in Chapter V. Figure 9-1 shows GT-VITA's OFMspert architecture tailored to the GT-POCC domain. The pedagogical design and architecture proposed in Chapter IV are captured in the pedagogy module. That is, LessonObject provides the structure for each lesson type and a PedagogyModule object provides the control for these lesson objects. In this section, the focus is on the flexibility of GT-VITA which lies in its explicit representation of domain knowledge and lesson content. The two categories of representation are described next.

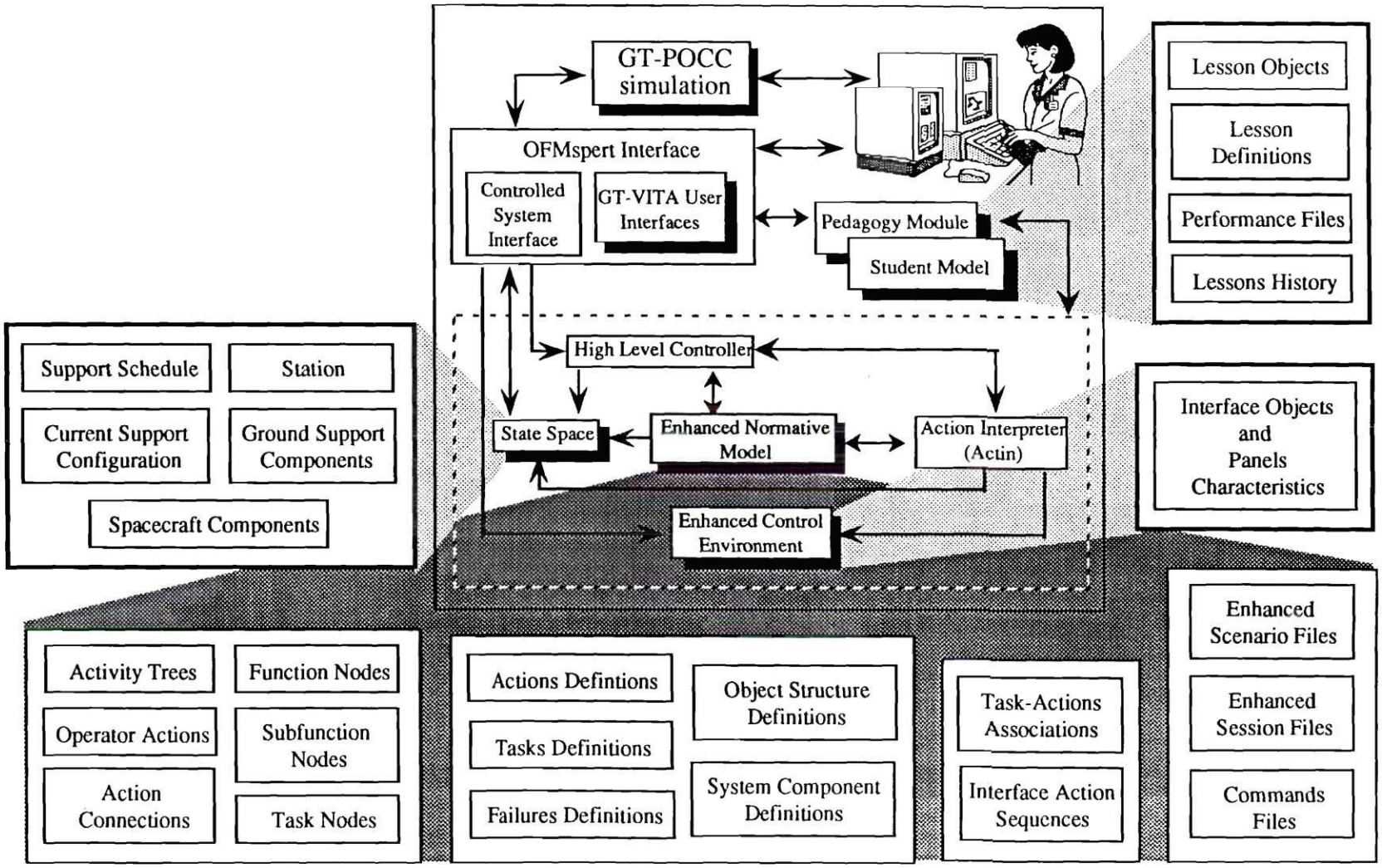


Figure 9.1 GT-VITA's OFMspert for intelligent tutoring

Domain Knowledge Files

Declarative knowledge in the form of explanations for actions, tasks, system failures and system components is structured in dedicated files from which GT-VITA accesses dynamically during tutoring. Tables 9-1 to 9-5 are examples of declarative files. Procedural knowledge in the form of action sequences for various tasks as derived from the operator function model is also structured in files. Figures 9-6 to 9-7 are example of procedural files. The session and scenario files described in Chapter VI are enhanced in GT-VITA to include instructional information. The first lesson to be initiated at the start of a session is added to the session file. Furthermore, for every scenario file, a corresponding commands file is specified which represents the set of operator commands that GT-VITA can emulate. Along with the set of commands are explanations about the commands that are used for instruction. Similarly, the scenario file is enhanced with explanations about the support scenario and the events to be simulated in general. Examples of session, scenario and commands files are depicted in Tables 9-8 to 9-10.

Table 9-1. GT-VITA's actions file that defines every action node of GT-POCC operator function model.

```
APSET
    APSET is the command to set up
    the Applications Processor (AP) for
    this support.
    You need to do this within two
    minutes before contact with
    the spacecraft.

NCCHEK
    NCCHEK is the command to
    check communications with the
    Network Control Center.
    You need to do this within two
    minutes before contact with
    the spacecraft.

DCI
    DCI is the command to
    inhibit Doppler compensation
    for two-way (coherent) tracking.
    You need to do this within two
    minutes before contact with
    the spacecraft.
    .
    .
    .
```

Table 9-2. GT-VITA's tasks file that defines every task node of the GT-POCC operator function model.

```
XC_TAC
.
*** XC_TAC
  Execute the configuration of the Telemetry and Command Computer
  that is needed for the upcoming support. The Telemetry and Command
  Computer must be properly initialized so that incoming telemetry
  flow and outgoing spacecraft commands are processed correctly
  during a real-time support. The XC_TAC task should be done as soon
  as the application processor has been initialized.


V_NCC
.
*** V_NCC
  Verify communications with the Network Control Center (NCC).
  During a real-time support the NCC monitors and analyzes
  communication network performances.  When communication between NCC
  and the MOR is established, feedback on overall network performance
  becomes available to you in the form of electronic
  Operational Data Messages (ODMs) throughout the real-time support.
  More importantly, NCC mediates communication between Goddard Space
  Flight Center and White Sands during TDRSS supports. Any request to
  reconfigure ground communications with the spacecraft (e.g.,
  reacquire lost spacecraft contact) are sent to and processed at NCC
  as a Ground Configuration Message Request (GCMR).  The V_NCC task
  should be done within 2 minutes before acquisition of signal (AOS).


MSOCC
.
*** MSOCC
  Monitor communications with the MultiSatellite Operations Control
  Center (MSOCC).  MSOCC provides the equipment to process telemetry
  data sent from the spacecraft and operator commands sent to the
  spacecraft.  In order that you can successfully monitor the
  spacecraft's health and safety and also the science data collection
  activities, you must monitor the quantity and quality of
  data arriving and leaving the Missions Operations Room (MOR). You
  should continuously perform the MSOCC task throughout the duration
  of the spacecraft contact.

    .
    .
    .
```

Table 9-3. GT-VITA's failures file which defines every GT-POCC failure event in terms of its effects, operator actions to take, and location of the failure.

```
(CSM1 makeAnomaly)
(CSM_cnt1)
.
*** Problem: There's an anomaly (flipped bit) in spacecraft
  command storage memory (CSM).
.
 Effects/Symptoms: Anomalies may compromise scientific data
   collection and the spacecraft's health and safety. The
   Events page flags that the CSM dump procedure is unsuccessful.
   The validation printout listing confirms the failed locations.
 Actions: File an anomaly report. Based on the failed locations,
   re-uplink the appropriate block to the CSM and re-verify the
   memory contents.
 Panels: (Command Storage Memory)


(TAC1 A1 hwFail) (TAC2 A1 hwFail) (TAC3 A1 hwFail)
(tac)
.
*** Problem: The A1 channel of the Telemetry and Command Computer has
  a hardware failure.
.
 Effects/Symptoms: Communications with the NCC may be blocked.
   Watch for the red outline of the A1 channel of the Telemetry and
   Command Computer.
 Action: Alert operator at the Data Operations Control System (DOCS)
   to fix the problem.
 Panels: (MSOCC) (Telemetry and Command Computer)


(BAT1 charge) (BAT2 charge)
(power)
.
*** Problem: A battery charge value is abnormal.
.
 Effects/Symptoms: Watch for the charge value to be yellow or red
   for a significant amount of time. Yellow means marginal and
   red means out of limits.
.
     red          yellow         green        yellow          red
 |------------|-------------|----------|--------------|------------|
    out of  marginal       normal       marginal      out of limits
  limits low      low                      high            high
.
 Action: File a report and check the mission operations document
   to see who to contact.
 Panels: (Spacecraft Power Subsystem)
   .
   .
   .
```

Table 9-4. GT-VITA's objects file that specifies the characteristics of interface objects and panels.

Panel name	Object name	Object label	Object type & value		Next panel	Definition file
system	spacecraft	(Spacecraft)	real	1.0	SCsystem	s/c.class
system	tdrs	(TDRSS)	real	1.0	tdrsSupp	tdrss.class
gsfc	nascom	(NASCOM)	string	NULL	NULL	nascom.class
gsfc	msocc	(MSOCC)	string	NULL	msocc	msocc.class
gsfc	ncc	(NCC)	string	NULL	ncc_cntl	ncc.class
tdrsSupp	wsc	(White Sands)	string	NULL	NULL	gn.class
SCsystem	ps	(Power Subsystem)	real	1.0	power	ps.class
ac	x_gyro	(X Gyro)	real	1.0	NULL	gyro.class
CSM_cntl	block_title	(Block Memory)	string	NULL	NULL	csm.class
rf	transponder1	(Transponder 1)	real	5.0	xpl_cnt1	xp.class
cdh	tape_recorder2	(Tape Recorder 2)	real	2.0	tr_cntl2	tr.class
.						
.						
.						

Table 9-5. Portion of GT-VITA's components file that defines system objects.

```
                ** Spacecraft Subsystems **

 This panel shows how a spacecraft is composed of many susbsystems
in addition to the science instruments which define its mission

            ** Command & Data Handling System **

    The command and data handling system (CDHS) is used to sample,
format, store and manage telemetry data (which consists of
scientific data and of spacecraft health and safety data). In
real-time operations, you can click on this object to view details
of CDHS.
                  ** Applications Processor **

 The Application Processor (AP), a critical part of the MSOCC,
contains software to monitor and control the spacecraft, process
telemetry data, display data in the Mission Operations Room (MOR),
log system events, and communicate with facilities external to
MSOCC.  MSOCC has three application processors, AP1, AP2 and AP3.

                      ** Battery **

  A Battery stores power to run the spacecraft. GASP has two main
batteries that supply power to all spacecraft subsystems.
   .
   .
   .
```

Table 9-6. GT-VITA's task-actions file that represents every task node with the corresponding sequence of commands to accomplish the task.

Table 9-6. GT-VITA's task-actions file that represents every task node with the corresponding sequence of commands to accomplish the task.

Task	Commands (or actions level of the GT-POCC OFM)
XC_TAC	(TACINIT)
XC_AP	(APSET)
V_NCC	(NCCHEK EVENTS)
MSOCC	(TACDISP EVENTS)
XSW	(SOFTWARE)
S/C	(PSDISP RFSDISP ACSDISP CDHSDISP TDRSSCHK EVENTS)
SCIENCE	(ERBECHK SAGECHK EVENTS)
DATA	(DATAACCT ACCOUNT)
XCSMVLD	(VLDVLD CSMVLD)
XEVENT	(EVREPORT REPORT)
XFWDURR	(FWDURR)
XDCI	(DCI)
XTERM_AP	(APTERM)
.	
.	
.	

Table 9-7. GT-VITA's action sequence file. For every group of operator commands, alternative sequences of actions are defined. Each action is represented by a panel, an object and an argument, if any.

Command	(TAC1 init) (TAC2 init) (TAC3 init)
Action Sequence 1	(system gsfc) (gsfc msocc) (msocc labelTac) (tac_cntl commands initialize) (tac_cntl okay)
Action Sequence 2	(tdrsSupp msocc) (msocc labelTac) (tac_cntl commands initialize) (tac_cntl okay)
Action Sequence 3	(GN_supp msocc) (msocc labelTac) (tac_cntl commands initialize) (tac_cntl okay) (TR1 mode playback) (system spacecraft) (SCsystem cdhs) (cdh tape_recorder1) (tr_cntl1 mode playback) (tr_cntl1 okay) (tdrsSupp spacecraft) (SCsystem cdhs) (cdh tape_recorder1) (tr_cntl1 mode playback) (tr_cntl1 okay) (GN_supp spacecraft) (SCsystem cdhs) (cdh tape_recorder1) (tr_cntl1 mode playback) (tr_cntl1 okay) . . .

Table 9-8. Example of GT-VITA's session file.

```
2 50 262 13 17 58
130  scenario1d   csmFile.S2  commands1.2a lesson1
580  scenario3.2  NULL        commands3.1
1090 scenario4.2  NULL        commands4.1
1660 scenario1.2  NULL        commands1.1
3550 STOP NULL NULL
```

Table 9-9. Example of GT-VITA's scenario file.

```
1 XP1  NT1  TDW  NAS2  TAC2  AP1  WS/NGT 300 12054
128  32  coherent  FAF320 i q
NULL NULL NULL NULL
61 TAC2 A1 hwFail
100 GYROY angle failOutOfLimitsHigh
182 TAC2 A1 hwFix
162 TDW loseLockOnRtnSignal
179 TAC2 B1 swFail
240 TAC2 B1 swFix
251 BAT2 cdRatio failMarginalLow
262 TDW acquireLockOnRtnSignal
289 BAT2 cdRatio setToNormal
291 GYROY angle setToNormal

.
**** This support uses the TDRS West Station (TDW) to communicate
 with GASP's Transponder 1 (XP2) and Antenna 1 (NT2).  XP2 is set
 to coherent mode for two-way data tracking.  There is no tape
 recorder playback or memory load operations scheduled for this
 support.  Spacecraft telemetry data will arrive at the Goddard Space
 Flight Center via White Sands through the "i" telemetry data stream
 at 128 kbps.  The "q" data stream channel is preambled at 6 kbps but
 no data are planned to be transmitted through it.  In terms of ground
 support, besides the Network Control Center, the NASA Communication
 Line 2 (NAS2), the Telemetry and Command Computer 2 (TAC2) and the
 Application Processor 1 (AP1) are needed.  The spacecraft contact
 is to last 5 minutes and your main goal for this support is to
 monitor the spacecraft's health and safety, including the scientific
 instruments on board.

.
**** The set of failures expected for this support are a mix of
 ground equipment failures and spacecraft parameter failures.
```

Table 9-10. Example of GT-VITA's commands file

```
-90 AP1 init
-80 TAC1 init
-53 NCC request odms
-42 NCC dci
90 TR1 mode standby
140 TR1 mode playback
319 TR1 mode record
431 AP1 terminated

.
*** For this lesson, the tutor will show you the commands that
 should be executed BEFORE the start of the support to configure
 and verify the necessary communications and equipment. That is,
 you learn about pre-pass activities.  The tutor will also show
 you the commands to execute a spacecraft tape recorder playback
 DURING the support.

.
*** Within two minutes before the start of the support, the
 tutor is going to configure first the application processor
 followed by the telemetry and command computer.  Then the
 tutor will verify communications with Network Control Center
 (NCC) by requesting Operational Data Messages (ODMs).  Since
 the transponder used for this support is in a coherent mode
 to enable two-way data tracking, the tutor will also request
 NCC to inhibit doppler compensation.
.
 Within the duration of the support, the tutor will command one
 of the tape recorder onboard the spacecraft to start playback.
 However, before a tape recorder can change from a record mode
 to a playback mode, the tutor must command it to go into the
 standby mode.  When the data playback is completed, the tape
 recorder is in a standby mode.  In order that data collection
 activities resume on the spacecraft, the tutor will command
 the tape recorder to start recording again.
.
 Shortly after the end of the support, the tutor will
 terminate the application processor so that it can be freed
 for other spacecraft use.
```

Lesson Files

The content of a lesson is structured in a file for a specific lesson type and processed dynamically by the corresponding LessonObject instanciated in the PedagogyModule. A lesson file represents feature information common to all lesson types and also information specific to a lesson type. Figure 9-2 shows the common portion of a lesson file. A lesson is characterized by its tutoring mode (e.g., Proactive), knowledge level (e.g., Declarative) and a lesson identification code (e.g., "1c"). The lesson file also specifies two parameters that are used for transition from one lesson to another: an integer representing the level of difficulty of the lesson and the file name of the next lesson to continue. Currently, the level of difficulty is not used; GT-VITA moves from one lesson to another either by repeating the current lesson or continuing to the next lesson. The goals of the lesson are also represented in the lesson file followed by instructions to traverse through the lesson. For most lesson types, the instructions for starting and halting the support scenario are also represented in the file.

Besides the common characteristics, a different lesson file is structured for every lesson type. The implementation of each lesson type is described below in terms how the lesson content is obtained.

```
                    Lesson X

TutorMode KnowledgeMode (X) LevelOfDifficulty NextLesson
Goal Explanations
Instructions
Scenario Explanations
```

Figure 9-2. Top portion of GT-VITA's lesson file structure

Learning System Components. The lesson file specifies the sequence of target panels and the corresponding objects for each target panel. Object explanations are accessed dynamically from system component files. Table 9-11 shows a sample lesson file of this type.

Learning System Behavior. Explanations for new concepts and various data flows are included in the lesson file. Other sources of information are obtained dynamically from the scenario and failure files. Table 9-12 is an example.

Exploring Tutor's Knowledge. The lesson file does not contain any information other than the common characteristics. The GT-POCC object structural definitions are stored in a readable form. These object representations are processed dynamically during lessons of this type.

Learning Operations by Example. The set of commands that GT-VITA plans to demonstrates are specified by the commands files associated with the current support scenario. The nature of the commands and the tutor's intentions are accessed from the commands file. The description of each command and the associated action sequence to accomplish the command are obtained respectively from the actions file and action sequences file. Thus, no specific information is necessary in the lesson file.

Learning Operations by Doing. The sequence of operator tasks to instruct is specified in the lesson file. The explanation of each task is obtained from the tasks file. The checklist of commands to accomplish the task is obtained from the task-actions file which represents the sequence of commands associated with each operator task. The action sequence of panels and objects for each command is dynamically read in from the action sequences file and displayed to the student in a help panel. The timing of each task to be instructed is determined by when the corresponding operator activity is hypothesized by ACTIN, the blackboard intent inferencer. Table 9-13 shows a sample lesson file of this type. The "NO" next to each task means that problems associated with the task are not instructed. Currently, GT-VITA does not have a "YES" option implemented.

Practicing Operations with Feedback/Checkpoints. For the practice lessons, no additional information is specified in the lesson files. These lessons interact extensively with the blackboard for feedback structures of activity assessments performed throughout the real-time support scenario. Other dynamic sources of helpful information come from the action sequences and actions files.

This concludes a brief discussion on GT-VITA's implementation in terms of the file-driven approach to knowledge and lesson representation. Structural definitions of the tutoring-related components of GT-VITA can be found in Appendix A. The dynamics of the blackboard (i.e., posting activity trees and connecting operator actions) are also implemented by a file-driven approach with structures adopted from Chronister (1990).

Table 9-11. Example of GT-VITA's lesson of type *Learning System Components.*

```
Proactive Declarative (1) 1 lesson1a
.
*** You will learn about the various components of the NASA
 data/information system.  In particular, you will become
 familiarized with the stations, equipment and facilities that
 support a spacecraft during a real time contact.


.
*** Components of the NASA information/data system are represented
 as objects in windows (or panels) to show what they do and how they
 relate to each other.
.
 For each panel that is shown, you are required to:
 1. study the general description of the target panel
 2. study the description of every item that is listed in the
    [Object List].

system 1
spacecraft wsc ground_network tdrs gsfc


.
                ** NASA Data/Information System **
.
   This panel shows the overall network of objects that make up the
 NASA satellite ground control system. Periodically, the spacecraft
 is scheduled to transmit data to the ground, where it is captured
 and analyzed.  These data are termed "telemetry" data.  Each
 instance of a scheduled data transmission is called a "pass" or
 "real-time support".  The data paths between the spacecraft and
 ground components are represented by the links between objects on
 this panel which are animated during a real-time support.

gsfc 0 system
nascom msocc ncc sdpf


.
                  ** Goddard Space Flight Center **
.
   The Goddard Space Flight Center (GSFC) houses various equipment
 and facilities that provide the ground support for the spacecraft

msocc 1 system
tac ap rup nascom


.
            ** Multisatellite Operations Control Center **
.
  The Multisatellite Operations Control Center (MSOCC) houses a
 network of computers and control centers. The heart of MSOCC is the
 Payload Operations Control Center (POCC) which is responsible for
 telemetry processing, command management, and interfacing with other
 facilities.
```

Table 9-12. Example of GT-VITA's lesson of type *Learning System Behavior.*

```
Modeling Declarative (2) 1 lesson3
.
*** You will learn about the dynamics of the NASA data/information
 system. Three concepts are introduced in this lesson.
 1. Support configuration
 2. Real-time data flow
 3. System failures

.
*** Things to look for:
 1. Elements of support configuration in various panels and objects
 2. Real-time data flow behavior as pointed out in the flow
    annotations
 3. Failures -- their location on panels and objects, their effects
    on data flow to other objects in the system.

.
*** Support Configuration
   A spacecraft contact is scheduled in advance to establish
 communication link with a TDRS station or a ground network. If TDRSS
 is used, a ground station will be needed for data transmission to and
 from the relay system.  Besides the station choice, the Support
 Configuration specifies the ground equipment and facilities that will
 be used during the scheduled contact. Also, a support is
 characterized by the spacecraft's transponder mode used in tracking
 data, and the various data transmission rates between the spacecraft
 and the ground.
.
*** Real-Time Data Flow
   During a real-time support, telemetry data is transmitted from the
 spacecraft to the ground via the return link signal. The data consist
 of both scientific data and health and safety data.
   .
   .
   .
*** System Failures
   There are three categories of failures the NASA data/information
 system may experience that may affect the activities of a real-time
 support or the spacecraft's mission in general.
   .
   .
   .
```

Table 9-12. Example of GT-VITA's lesson of type *Learning System Behavior* (cont'd).

```
*** To learn about data flow and system failures, the lesson scenario
 should be run in real time. You can freeze the scenario at any time
 to help you view particular system states or behaviors more closely.

****
   During a TDRSS support, telemetry data are transmitted continuously
 from the spacecraft to White Sands ground station via one
 of the two relay systems -- TDRS West or TDRS East.  White Sands then
 transmits the data to Goddard Space Flight Center. During a Ground
 Network support, telemetry data are transmitted from the spacecraft
 to a ground network, and subsequently to GSFC. Spacecraft commands
 issued by the FOT analyst during the support is transmitted out of
 GSFC and eventually to the spacecraft.
.
    .
    .
    .
****
   At Goddard Space Flight Center, telemetry data enter the
 MultiSatellite Control Center (MSOCC) and spacecraft commands leaves
 MSOCC via the NASA communication (NASCOM) lines.
    .
    .
    .
****
   The spacecraft's mission is defined by the scientific instruments
 onboard that conduct experiments and collect scientific data. GASP
 (the spacecraft) has two science instruments, SAGE and ERBE. The data
 collected are processed by computers called Telemetry and Command
 Units (TCU1, TCU2) and stored temporarily on tape recorders
 (TR1, TR2). Although the instruments are stand alone spacecraft
 components, they are included within the Command and Data Handling
 Subsystem to show the data collection cycle. This data collection
 cycle is represented by the two grey arrows between an instrument,
 a TCU and a TR.  The data collection process goes on continuously;
 the data path is animated (by green arrows) regardless of any real
 time contact.
.
    .
    .
    .
```

Table 9-13. Example of GT-VITA's lesson of type *Learning Operations by Doing*.

```
Proactive Procedural (5) 1 lesson5a
.
            ****** LEARNING OPERATIONS BY DOING **********
.
*** In this lesson you will learn to execute the various tasks to
 configure and verify the necessary communications and equipment
 BEFORE the start of the support.  Other setup activities include
 performing pre-pass bookkeeping. Specifically, you will learn the
 following tasks to support the configuration, verification and
 setup functions:-
   Configuration:  XC_AP, XC_TAC
   Verification:    V_NCC, XSW, XSCHED, XCSM
   Setup:          XDCI, XPLAN
 Each task requires one or more command steps to complete.  Your
 goal is to become proficient in the set of commands involved in
 these pre-pass activities.


.
* When the tutor alerts you to perform a task (with a beep), click
    on the [Task Explanation] button to get detailed explanation of
    the task being taught.
* Next, click on the [Steps in Task] to find out the command steps
    associated with the task and the [Current Step] that you should
    attend to.
* Check the [Help] box of the current step to get the action sequence
    the command requires and explanation of the current step.  Carry
    out the action sequence as indicated to complete the step. The
    tutor requires that your actions match those suggested on the
    help panel. Complete all steps in a task the same way.
* When you have completed a task, click on [Continue] when you are
    ready for the next task. Observe the effects of your actions
    on system states.
* In between tasks, you can explore system behavior and system
    failures freely.


.
*** This lesson scenario must be run in real time. When a task is
 ready to be performed, the tutor pauses the scenario so that you
 have the opportunity to study the task and carry out the steps
 associated with the task.  The scenario does not restart until you
 are ready to move on to the next task.  Unless a task is pending
 to be executed, you can pause and restart the scenario anytime
 during the support.

XC_AP   NO
XC_TAC  NO
V_NCC   NO
XSW     NO
XSCHED  NO
XCSM    NO
XPLAN1  NO
XDCI    NO
```

GT-VITA: The Tutor/Aid Paradigm Revisited

As a proof-of-concept ITS, how well does GT-VITA manifest the tenets of the tutor/aid paradigm presented in this thesis? In this section, GT-VITA is discussed with respect to the proposed characteristics of domain knowledge, the pedagogical design, and knowledge of the student (See Table 3-1, Figure 3-1 and Table 3-2).

GT-VITA's Domain Knowledge

Overall, GT-VITA represents and communicates the GT-POCC domain in accordance to the characteristics proposed by the tutor/aid paradigm. Based on the need assessment phase as discussed in Chapter VI, GT-VITA's domain knowledge is *complete* from the viewpoint of a FOT analyst. That is, the GT-VITA and GT-POCC systems combine to capture as much knowledge about the NASA system as a FOT analyst needs to know. The domain knowledge is distributed among the GT-POCC simulation, GT-VITA OFMspert's enhanced normative model, state space and control environment.

The graphical representation of GT-POCC provides a *relevant* form to visualize concepts such as data flow and object relations. The graphical representation also reflects the *hierarchical* decomposition of system components with *inspectable* objects. Moreover, GT-VITA's internal hierarchical representation of objects is also inspectable through lessons of type *Exploring Tutor's Knowledge*. Furthermore, the operator function model as dynamically represented on the inspectable blackboard structures operator activities in a relevant and hierarchical form.

GT-VITA's graphical and interactive environment enables a student to view the operator task from a "big picture", thus developing useful mental models about the NASA system. This environment structured in lessons encourages self-pacing and exploring, and provides the practice environment to coordinate various FOT activities; that is, GT-VITA supports the development of high-level skills. Finally, the progression of the proposed lesson types communicates the domain knowledge from form/structure to functions to operations.

GT-VITA's Pedagogical Knowledge

Overall, GT-VITA's pedagogical knowledge is structured with the pedagogical design of lesson types and the architecture of lesson objects as proposed by the tutor/aid paradigm. Currently, except for the last lesson type, *Performing Operations with Tutor as Aid*, GT-VITA has the knowledge of all lesson types. These lesson types capture GT-VITA's capability to play various tutoring roles for the three dimensions of knowledge. GT-VITA can be *proactive, reactive* or be in a limited *coaching* mode. GT-VITA provides *empowering tools* and demonstrates (*models*) system behavior and operator activities. Throughout the lessons, GT-VITA provides *help* with instructions and alert messages. On most lesson types, students can also access help on object definitions by clicking on the help button on the top right hand corner of every panel as symbolized by "?".

Since the transition from tutor to aid is not implemented for GT-VITA, GT-VITA does not have the capability to allow students to delegate various tasks to GT-VITA. That is, GT-VITA does not function as an *assistance* yet, even though it has the knowledge to perform most operator activities.

GT-VITA also adheres to the six general principles proposed for the pedagogical design. Basically, lesson goals are made explicit; reactive exploratory learning, self-pacing and self-monitoring are encouraged; all lesson types share common features such as the lesson panel, explanation panels and pause/restart option; all goals are inferred as accomplished when a lesson ends; and finally, most FOT activities are manifested through visible objects.

GT-VITA's Knowledge of the Student

GT-VITA's knowledge of the student is represented within the various lesson objects and the blackboard structure of intent inferencing. An overlay of the operator function model and the enhanced declarative and procedural knowledge within the Enhanced Normative Model serve to define the goals of a lesson and its lesson content. Common errors that a student encounters in interacting with the tutor and/or the system are captured in each lesson object. Thus, GT-VITA has knowledge about correct and incorrent student actions.

Each lesson object has communication knowledge to process student actions of three types: DIALOG, TARGET and CONTROL. This explicit encoding enables GT-VITA to opportunistically diagnose student actions as they happen within each lesson type. Similarly, during practice lesson types, ACTIN also interprets student actions based on the best hypothesis of operator activities at the time. More importantly, GT-VITA can be sensitive to the situatedness of each student action and student-tutor interactions due to the encapsulation of communication knowledge in each lesson object. In other words, what the student can or cannot do in any situation and how the student interacts with the tutor are determined by the instructional goals of a lesson type.

Each lesson type has its own performance measures and they are recorded from lesson to lesson. However, at present, GT-VITA does not have an explicit competence model of the student. The tranisition from lesson to lesson is determined manually by the human supervisor. As a result, the progression of lesson types assume overall student competence at the end of training. At present, data concerning student performance within lessons and between sessions are recorded but not integrated with the diagnostic methods in a lesson object.

In summary, GT-VITA has been implemented to closely encompass the tenets of the tutor/aid paradigm within the scope of this research. In the next chapter, an evaluation study conducted for GT-VITA to examine the effectiveness of the tutoring aspect of the tutor/aid paradigm is described.

CHAPTER X

EVALUATION OF GT-VITA

A systematic approach to training involves three phases: need assessment, training development, and evaluation (Goldstein, 1986). The tutor/aid paradigm proposes characteristics and pedagogical design of ITSs for complex domains (Chapters I through VI). As applied to the domain of satellite ground control, this thesis also identifies specific knowledge requirements for a novice operator in that domain (Chapter VI). Subsequently, GT-VITA is developed to capture the tutoring end of the tutor/aid paradigm in the GT-POCC environment (Chapters VII, VIII and IX). Thus, this thesis has so far covered first two of the three phases: assessing the instructional needs for a complex domain and developing the training environment to meet those needs. The final phase - evaluation - is the subject of this chapter.

Goldstein (1986) defines evaluation as "the systematic collection of descriptive and judgmental information necessary to make effective training decisions related to the selection, adoption, value, and modification of various instructional activities" (p. 111). In other words, as cautioned by Goldstein, evaluation is more than labeling a training program as good or bad; rather, information gathered from the evaluation process should be dynamically utilized to further improve the program. The evaluation of GT-VITA is conducted in the same spirit. It serves as the first attempt to assess the the effectiveness of the tutor/aid paradigm. This chapter first identifies the objectives and scope of the evaluation. Then, the training process and the data collection procedures are presented.

Objectives and Scope of the GT-VITA Evaluation

The underlying goal of evaluation is to test the tutuoring validity and utility of the tutor/aid paradigm as implemented on the GT-VITA system. The tutor/aid paradigm characterizes an intelligent

tutoring and aiding system in terms of the domain knowledge to be communicated, the pedagogical matrix that defines "when and how to teach what", and the tutor's knowledge of the student. Furthermore, the tutor/aid paradigm proposes a pedagogical design that captures these characteristics in a coherent framework for training novice operators for the role of supervisory control in complex systems. As discussed in the previous chapter, GT-VITA does embrace the tutor/aid paradigm faithfully. So, the litmus test for GT-VITA is whether the characterization and pedagogical design of the tutor/aid paradigm yielded an intelligent tutoring system that effectively train novice operators to become competent operators in monitoring and controlling GT-POCC. Some of the questions that this evaluation addresses are:

1. Does GT-VITA appropriately teach the knowledge and skills that an FOT analyst should have?
2. Can a novice operator be trained in a reasonable amount of time?
3. Is GT-VITA sufficiently adaptive and flexible to accommodate individual differences among adult trainees?
4. How do potential users and training personnel react to GT-VITA?

What is *not* evaluated at this time is the extent to which GT-VITA transitions from a tutor to an aid, and how well operators trained on GT-VITA use GT-VITA as an aid. Because the transition lesson type, the last of the eight lesson types proposed (see Table 4-1), is not implemented in GT-VITA, issues about transitions is beyond the scope of the present evaluation process.

Participants of GT-VITA Evaluation

Inasmuch as GT-VITA is a prototype training system, it is designed specifically with novice FOT analysts at NASA in mind. As a result, a decision was made to evaluate GT-VITA with actual NASA personnel who qualify to be novice FOT analyst within a realistic context. There are several reasons such a decision becomes necessary.

First is the problem of subject selection. In the past, academic students have served as participants in evaluating prototype systems built for NASA (e.g., GT-MSOCC, Mitchell, 1987). Such evaluation

experiences have capitalized on the convenience and large number of subjects on site (i.e., at school), and the experimental control afforded by the artificial environment. Unfortunately, even though statistical results were very positive, the major complaint from NASA has been the lack of generalizability of these results to real world operators (Mitchell, 1991). The student subjects versus actual operators were motivated differently. Frequently, student subjects were given monetary rewards or bonus points in a course for participation. Whereas, actual operators participate if the system being evaluated relates to their job and could affect their performance on the job. However, further evaluations of, or future research for GT-VITA should capitalize on the convenience, experimental control, homogeneity, and sample size afforded by academic subjects.

Therefore, a commitment was made for GT-VITA to avoid this pitfall in subject selection; so realism won over at the expense of experimental control in the pursuit of *scientifically significant* results (Goldstein, 1986). The concept of scientific significance is "... the establishment of meaningful results that permit generalizations about training procedures beyond the immediate setting being investigated" (p. 148). Scientific significance concerns not just with statistical significance but also with practical significance -- the extent the training treatment yields reliable and recurring changes.

Second is the issue of trainee readiness for the instructional program. Trainee readiness is the maturational and experiential background of the potential trainees (Goldstein, 1986). GT-VITA is developed for technically oriented adults who assume minimal background knowledge about the mission control activities at NASA; GT-VITA does not instruct NASA concepts from "square one". Even though academic students share many common factors such as educational level and age, they do no share any working knowledge about NASA that is used within their environment on a daily basis. The reverse is generally true for NASA personnel who may differ in background and age, but they share some working knowledge about NASA. Thus, the better trainee readiness for NASA personnel is more an advantage than the potential lack of homogeneity among them is a disadvantage.

Third, as a first attempt to evaluate GT-VITA, it is believed that comments and reactions from NASA personnel will be more meaningful than those from academic subjects for the same reasons about trainee motivation and readiness.

Evaluation Design

The evaluation of GT-VITA was conducted at NASA Goddard Space Flight Center with NASA personnel who have no on-the-job experience as a FOT analyst in the ERBS or COBE mission control environment which GT-POCC simulates. Permission was granted by NASA officials to complete the evaluation process in less than two weeks. Consequently, potential participants were asked to volunteer an hour a day for about seven work days.

The goal of this first evaluation study was maximum participant exposure to GT-VITA. As a result, the evaluation procedure is a pre-experimental design with no control condition (Goldstein, 1986); every subject is trained on GT-VITA. The training process was more accelerated than if GT-VITA was incorporated with an integrated training program. The focus was for everybody to go through every lesson type and be reasonably competent in controlling the GT-VITA interface at the end of training. No lessons for training the GT-POCC task interface were given.

Participant Profile

The evaluation was carried out within the Mission Operations Division at NASA Goddard Space Flight Center. With the commitment to evaluation within a real-world context, the goal was to achieve maximum exposure of GT-VITA to gather data, input and feedback from different perspectives. Thus, for a preliminary and overall GT-VITA evaluation, the focus was to have participants representing a cross section of the Missions Operations Division, instead of any one group in particular. Ultimately, such an arrangement was deemed to produce a more meaningful and interesting evaluation experience.

Voluntary participants were pooled from four groups in Mission Operations; each with a different role at NASA. The fours groups were NASA government personnel, computer operators, spacecraft specialists and software testing team. Thus, each group was interested in GT-VITA for different, but complementary reasons. The groups also differed in their knowledge and experience as an FOT analyst, and in their computer literary such as familiarity in using the mouse input device. All participants qualified as novice FOT analyst in the evaluation since no one had FOT analyst experience in a GT-POCC-like

environment. Each group is described below in terms of the motivation and general profile of the participants from the group.

NASA Mission Operations Division Personnel (Group A)

The motivation for this group was the technology transfer of research ideas to address existing problems in mission operations, and for the design of future mission control systems. There were four participants from this group. Subject A1 holds a top managerial position. Subject A2 is a project manager. Subject A3 is a cognitive scientist on a research team. Subject A4 is a human factors specialist. None of the subjects had any prior experience as a FOT analyst, but all had general knowledge about the duties of an FOT analyst. Except subject A1, all were computer literate. Subject A1 had never used a mouse input device prior to GT-VITA.

NASA Mission Operations Computer Operators (Group B)

This group provides the ground equipment support for mission operations of all existing satellites. The three computer operators who participated were enthusiastic about the opportunity to be trained on GT-VITA. They perceived the knowledge and experience gained from the FOT training to be potentially beneficial in future job advancements. Subject B1 was an operator at the Multi-Satellite Operations Control Center (MSOCC); subjects B2 and B3 were controllers at the Data Operations Center (DOC) inside the MSOCC facility. Both DOC controllers were former MSOCC operators who advanced with experience and skills. The three subjects had no knowledge of FOT operations and their knowledge of computers was limited to what they do on the job. In fact, Subjects B1 and B2 had never used a mouse device.

General Electric (NASA Contractor) Spacecraft Specialists (Group C)

This group is directly involved in the design of future spacecrafts, payload operations and mission control systems, and in the preparation of FOT analysts for future mission operations. Thus, group participants were interested in the GT-VITA training process, and the approach and methods in designing GT-VITA. Subject C1 was a mission operations manager. Subject C2 was a spacecraft and instruments

specialist. Subject C3 was a specialist in spacecraft subsystems hardware. Both subjects C1 and C2 had many years of FOT experience in earlier generations of spacecrafts. Subject C3 had no prior FOT experience at all. All subjects were comfortable with interactive computer systems.

Bendix (NASA Contractor) FOT Software Testing Team (Group D)

The software team is responsible for testing and verifying software used by FOT analysts for mission control. Team members have general knowledge but lack the experience of FOT operations. As a result, the opportunity to be trained on GT-VITA would provide an added edge to their job in software testing. Another motivation was purely software oriented; they were curious about how GT-VITA worked and were eager to "break" the system. Naturally, all of them were proficient in computer interactions. Four team members were able to participate (subjects D1, D2, D3 and D4), and due to a scheduling problem, the subjects were trained in pairs; more about the training process is presented in the next section.

Two other participants were included in this group even though they were not actually software team members. Subjects D5 and D6 were part-time programmers in a spacecraft related research project. There were two reasons for considering subjects D5 and D6 here. First, they shared similar motivational goals for participating in the GT-VITA training. Second, the subjects were also trained as a pair.

There was a total of sixteen participants that yielded thirteen trainees; the six participants in Group D collapsed to three teams of two with each pair being trained concurrently. The training process for each trainee is discussed next.

The Training Process

The training environment was set up in a vacant mission operations room at NASA Goddard Space Flight Center. GT-VITA was configured to operate as a stand-alone system on a Sun Sparc workstation with two 16-inch color monitors. Users interact with GT-VITA primarily with a mouse input device.

Before the training process started, all trainees scheduled sessions with the experimenter, an hour a day during their normal working hours for GT-VITA. Subjects from Groups A, C an D were trained during the day while subjects from Group B were trained during the midnight shift. In the first session, the experimenter introduced the purpose of the training process and briefed the trainee about general features of GT-VITA such as the lesson panel and mouse interactions. Instructions for operating each lesson type were available but not needed since the training program was supervised by the experimenter and conducted in an accelerated pace (see Appendix B).

Curriculum of GT-VITA

Basically, every participant was trained with all seven lesson types beginning from *Learning System Components* to *Practicing Operations with Checkpoints*. The total number of lessons completed and the number of days to completion varied among the participants according to each participant's pace and style of learning.

The curriculum of GT-VITA was planned in advance by the experimenter to cover the knowledge requirements identified for GT-POCC operator. The curriculum included both the declarative and procedural elements of GT-POCC. Declarative elements involved understanding of objects and relations of the NASA data/information system at various detailed levels, as shown in Table 10-1. Procedural elements involved understanding and performance of FOT operations in three categories: pass procedures, trouble management and spacecraft commanding, as shown in Table 10-2. The lesson types used to support the curriculum are discussed below in terms of the three phases of the GT-VITA training process: declarative, procedural and practice. The scope of the training process was to instruct all major aspects of the curriculum, and to foster efficient performance in the as many FOT operations as time permitted.

Table 10-1. Declarative components of GT-POCC knowledge

Components of NASA network
Spacecraft components
Ground support components
MSOCC components
Data flow between components
Hardware failures
Software failures
Spacecraft Support types
Support configuration

Table 10-2. Procedural components of GT-POCC knowledge

Pass Procedures
Pre-pass tasks
On-pass tasks
Post-pass tasks
Trouble Management
Ground component failures
Spacecraft abnormalities
Spacecraft signal problems
Spacecraft Commanding
Tape recorder data playback
Command storage memory load
Command storage memory dump

Declarative Phase

In the declarative phase, two lessons of type *Learning System Objects* were given to instruct system components: one focusing on ground support components of the NASA network, the other on the spacecraft and its subsystems. A lesson of type *Learning System Behavior* taught about data flow, failures and support configuration. This lesson could be repeated with different support scenarios. Originally, three support scenarios were planned to separately demonstrate spacecraft abnormalities, ground equipment problems, and communication failures. In the accelerated training mode, all three categories were demonstrated in one long support scenario. This phase ended with one lesson of type *Exploring Tutor's Knowledge.*

Procedural Phase

In the procedural phase of training, the activities to command a spacecraft tape recorder playback and a spacecraft memory load and dump were instructed with two lessons of type *Learning Operations by Example:* one for the playback operations and the other for the command storage memory operations. The lessons also demonstrated pre-pass and post-pass procedures. Two lessons of type *Learning Operations by Doing* were planned: one to instruct pre-pass, on-pass and post-pass activities, and one to instruct procedures to rectify various problems and failures during a support. At the time, there was no lesson of type *Learning Operations by Doing* planned for spacecraft commanding operations.

Practice Phase

The two lesson types, *Practicing Operations with Feedback* and *Practicing Operations with Checkpoints*, operated with practice scenarios that focused on pass procedures and trouble management aspects of the FOT operations. Since participants did not have the opportunity to perform commanding activities during the procedural phase, no support scenarios reflected the need for spacecraft commanding. Participants were considered trained if they could perform pass procedures and manage for trouble in a timely manner during real-time support scenarios. A printed copy of the FOT operations checklist was made

accessible during this phase of training for reference. The transition from the first practice lesson type to the second, and the number of practice lessons completed were determined by the participant's confidence in the FOT operations, and also on how much time was left in the training process.

Throughout the training process, various evaluation data were collected from participants or online as GT-VITA was running. Data collection and analysis are discussed next.

Data Collection and Analysis

Figure 10-1 summarizes the data collection points during the GT-VITA training process. In accordance with the objectives of the GT-VITA evaluation, three categories of data were collected: (1) questionnaires on participants' knowledge, (2) participants' evaluation of their own understanding and performance, (3) GT-VITA's objective performance assessment data, (4) participants' subjective evaluation of GT-VITA, and (5) reaction of NASA technical and training personnel to GT-VITA. Since there was no control condition, data analysis generally related to comparisons and differences among subject groups. Because of the small sample size for each group, nonparametric statistical inference procedures were used to avoid any false assumptions about normality and independence of the error term (Gibbons, 1985). The collection and analysis of each data category are described next in the chronological order.

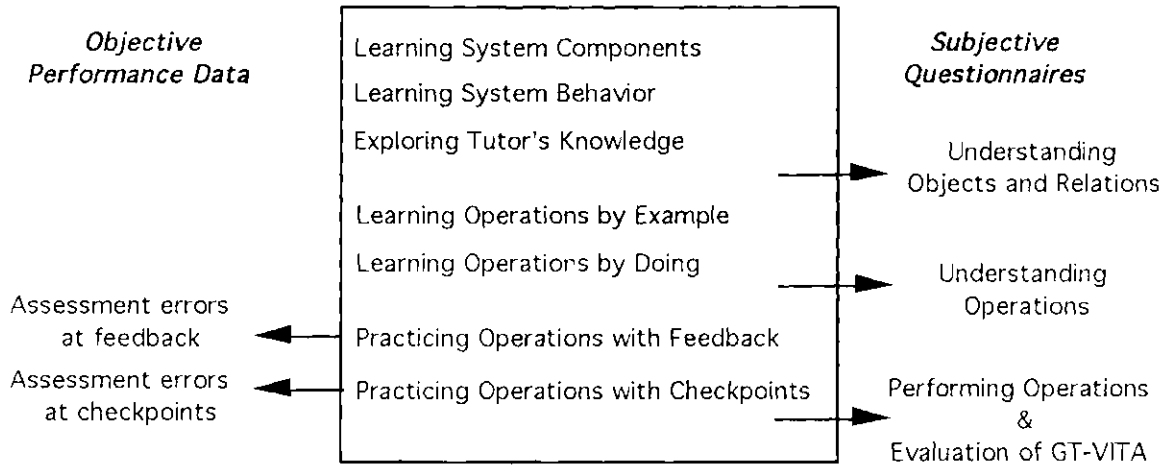


Figure 10-1. Collection points for GT-VITA evaluation data.

Participants' Knowledge Questionnaires

At the end of the declarative phase of training, the experimenter verbally questioned each subject on his or her understanding of the major objects and relations in GT-POCC. This questionnaire was a form of posttest to evaluate if the lessons administered so far helped the student learn the declarative elements of the GT-VITA curriculum. The questions focused on naming components at various level of details, identifying data flow paths between components, and explaining different failure types. Since the objective is not to test rote memory, but understanding, the student is encouraged to explore the GT-VITA interface for answers. Table 10-3 lists the questions asked by the experimenter.

Similarly, at the end of the procedural phase of training, the experimenter verbally questioned a subject on his or her understanding of the major FOT operations in GT-POCC. The questionnaire tested if the subject could identify major FOT operations to monitor a support, manage failures, and to command the spacecraft. Again, the subject could seek answers by exploring the GT-VITA interface. Table 10-4 lists the questions asked by the experimenter. Copies of these verbal questionnaires can be found in Appendix C.

Besides serving as posttests, data from this category were collected as checkpoints on the overall training process. Specifically, because the training was accelerated for evaluation, the experimenter wanted to ensure that the training was not too accelerated that nothing was learned!

Participants' Subjective Performance Evaluation

At the end of the declarative phase, and after the verbal evaluation, each subject filled in a questionnaire that gauged the subject's perceived understanding of the GT-POCC objects and relations. Basically, the students rated how strongly they agree or disagree with statements about their understanding of the declarative elements listed in Table 10-1. The Barlett rating scale (Goldstein, 1986) was used with five choices "Strongly agree", "Agree", "Neither agree nor disagree", "disagree", and "Strongly disagree" for all questionnaires in this category. Table 10-5 shows the questionnaire.

Similarly, at the end of the procedural phase, and after the verbal evaluation, a subject filled a questionnaire that gauged the subject's perceived understanding of FOT operations in GT-POCC. Students

rated statements constructed with the procedural elements listed in Table 10-2. Table 10-6 shows the questionnaire.

A final questionnaire was given at the end of the training process that asked subjects to rate their confidence in performing the various GT-POCC operations. At this time, students would have the opportunity to practice performing most of the FOT operations in the practice phase. Table 10-7 shows this part of the final questionnaire.

Table 10-3. Posttest questionnaire on GT-POCC objects and relations

1. What are the major components in the NASA Goddard information/data system?
2. What are the major components in the Goddard Space Flight Center (GSFC) ground control system?
3. What are the major components in MSOCC?
4. What are the major TAC components?
5. What are the major spacecraft components in the GT-POCC?
6. In the radio frequency spacecraft subsystem, what components do the FOT monitor?
7. In the power spacecraft subsystem, what components do the FOT monitor?
8. In the attitude spacecraft subsystem, what components do the FOT monitor?
9. In the command and data handling spacecraft subsystem, what components do the FOT monitor?
10. What are the two types of memory in a spacecraft??
11. Trace data flows within the spacecraft
12. How does the tutor interface represent a hardware failure?
13. How does the interface distinguish between hardware and software failures.
14. Distinguish between the flow of data during TDRSS versus ground network support.
15. Trace data flows within MSOCC and the TAC. Describe how a failure of a component affects data flows.
16. Consider the support configuration panel, describe the components that will support the ground control and identify the activities that will be carried out during the scheduled support.

Table 10-4. Posttest questionnaire on GT-POCC operations.

1. What are the major pre-pass FOT tasks for a real-time support?
2. What are the major on-pass FOT tasks for a real-time support?
3. What components of the ground system does the FOT monitor during a real-time support? What indicates a failure at a component?
4. What components of the spacecraft does the FOT monitor during a real-time support? What kinds of failures occur?
5. What steps are involved in a tape recorder dump?
6. What steps are involved in a CSM load?
7. What steps are involved in a CSM dump?
8. What are the steps involved in the forward and return link reacquisition after "loss of lock"?
9 . What are the major post-pass FOT tasks for a real-time support?

Table 10-5. Questionnaire on GT-POCC objects and relations

1. After using GT-VITA I now feel that I understand the major components of the NASA Goddard information/data system.
2. After using GT-VITA I now feel that I understand the major components of a typical spacecraft in the NASA Goddard information/data system.
3. After using GT-VITA I now feel that I understand the major components of the ground support system in the NASA Goddard information/data system.
4. After using GT-VITA I now feel that I understand the major components of the MultiSatellite Operations Control Center (MSOCC) in the NASA Goddard information/data system.
5. After using GT-VITA I now feel that I understand how data are transmitted within the spacecraft.
6. After using GT-VITA I now feel that I understand the effects of hardware failures in the NASA information/data system.
7. After using GT-VITA I now feel that I understand the effects of software failures in the NASA information/data system.
8. After using GT-VITA I now feel that I understand the difference between TDRSS and ground network supports.
9. After using GT-VITA I now feel that I understand support configuration requirements.

Table 10-6. Questionnaire on GT-POCC operations

1. After using GT-VITA, I now feel that I understand the pre-pass support tasks.
2. After using GT-VITA, I now feel that I understand the on-pass support tasks.
3. After using GT-VITA, I now feel that I understand the major failures in the ground system network and how they affect the flow of data and commands between the MOR and spacecraft.
4. After using GT-VITA, I now feel that I understand the major failures in the spacecraft and how they affect the flow of data and commands between the MOR and spacecraft..
5. After using GT-VITA, I now feel that I understand the tasks involved in a tape recorder dump.
6. After using GT-VITA, I now feel that I understand the tasks involved in a command storage memory load.
7. After using GT-VITA, I now feel that I understand the tasks involved in a command storage memory dump.
8. After using GT-VITA, I now feel that I understand the post-pass support tasks.
9. After using GT-VITA, I now feel that I understand the tasks involved in a re-acquisition of the signal for the forward and return links.

Table 10-7. Final questionnaire on GT-POCC operations

1. After using GT-VITA, I now feel confident that I could perform pre-pass support tasks.
2. After using GT-VITA, I now feel confident that I could perform on-pass support tasks.
3. After using GT-VITA, I now feel confident that I could manage failures in the ground system network.
4. After using GT-VITA, I now feel confident that I could manage failures in the spacecraft.
5. After using GT-VITA, I now feel confident that I could perform the tasks involved in a tape recorder dump.
6. After using GT-VITA, I now feel confident that I could perform the tasks involved in a command storage memory load.
7. After using GT-VITA, I now feel confident that I could perform the tasks involved in a command storage memory dump.
8. After using GT-VITA, I now feel confident that I could perform post-pass support tasks.
9. After using GT-VITA, I now feel confident that I could perform the tasks involved in a re-acquisition of the signal for the forward and return links.

GT-VITA's Objective Performance Assessment

Throughout the training process, the GT-VITA program generated many log files that contained potentially useful data including a history of student actions, a history of all system events, the student's reaction times for some operator activities, and all activity assessments conducted by GT-VITA during practice phase. For the current evaluation, only the activity assessment data were used. These data list the time an operator activity was assessed by ACTIN and the results of the assessment in terms of missing or repeated actions. These data were collected for every real-time support run under lessons of type *Practicing Operations with Feedback* and *Practicing Operations with Checkpoints.*

There was a total of eleven types of activities that were assessed during the practice scenarios used for the current GT-VITA practice training. Before the start of a pass, the configuration, verification, setup for Doppler compensation inhibition, and bookkeeping functions were assessed. The activities during a pass are monitoring, bookkeeping, and trouble management in terms of ground or communication failure management and bookkeeping, and spacecraft abnormalities bookkeeping. Terminating the application processor and bookkeeping are post-pass functions being assessed. Table 10-8 summarizes the activity assessments. Percentages of time each assessment was concluded unsatisfactory (either at feedback points or checkpoints) were calculated for each subject group. Freidman's two-way analysis of variance was used to examine if there were group differences in the commitment of assessment errors. Furthermore, the assessment error percentages were ranked for each group and the Kendall's coefficient of concordance was computed. This coefficient measures the extent to which a number of rankings agree. For GT-VITA evaluation, assuming that high percentage of an assessment error could be equated to a problematic or difficult activity, the coefficient could ascertain if there were group differences in what activities seemed problematic or difficult.

Table 10-8. GT-POCC activity assessments.

Pre-pass	CONFIG VERIFY DCI BOOK	Configuration of MSOCC equipment: AP and TAC Verification of AP software, CSM listings and support schedule. and communications with NCC Doppler Compensation Inhibition setup Pre-pass bookkeeping
On-pass	FAIL-MGT FAIL-BOOK ANOM-BOOK MONITOR BOOK	Management for ground component failures Bookkeeping for ground or communication failures Bookkeeping for spacecraft abnormalities Monitoring of spacecraft health and safety On-pass bookkeeping
Post-pass	TERM BOOK	Termination of AP Post-pass bookkeeping

Participant's Subjective Evaluation of GT-VITA

The final questionnaire given at the end of the training process was also designed to capture subjects' overall reaction to GT-VITA. One question asked subjects to rate how strongly they perceived GT-VITA as a useful tool for new FOT analysts. The second question asked subjects to rank each lesson type in GT-VITA in terms of its contribution to their overall knowledge gained from the training (1 = very little, 10 = greatly). The last question asked subjects to rank each lesson type in terms of its potential usefulness if GT-VITA was fully implemented and integrated in FOT training (1 = not useful, 10 = extremely useful). The last question was intended to evaluate GT-VITA outside the experimental artifacts such as accelerated training and unforeseen computer problems. These questions are shown in Table 10-9.

Freidman's analysis of variance and Kendall's coefficient of concordance computation were performed on the rankings of lesson types for the four subject groups. Two questions were asked: did the groups rank the lessons equally, and did ranking between lessons differ among groups?

Table 10-9. Final questionnaire on GT-VITA

10. Overall, I think GT-VITA is a useful tool for tutoring new Flight Operations Team analysts.

11. For each type of GT-VITA lesson given below, rate how much it contributed to the overall knowledge you gained from GT-VITA on a scale from 1 to 10 (1 = contributed nothing or very little, 10 = contributed greatly). Also, feel free to note any comments or suggestions.

12. For each type of GT-VITA lesson given below, rate its potential usefulness as a fully-implemented portion of FOT training on a scale from 1 to 10 (1 = if fully implemented, would not be useful at all, 10 = if fully implemented, would be extremely useful). Also, feel free to note any comments or suggestions.

Learning System Components _____
Learning System Behavior _____
Exploring the Tutor's Knowledge _____
Learning Operations by Example _____
Learning Operations by Doing _____
Practicing Operations with Feedback and Checkpoints _____
Practicing Operations with Checkpoints _____

Besides the four categories of data collected, verbal comments from participants and the experimenter's observations were also noted. In summary, this chapter detailed the approach to the first attempt to evaluate GT-VITA. The hope was to yield meaningful results in a real-world context. All these results are reported and discussed in the next chapter.

CHAPTER XI

RESULTS OF EVALUATION OF GT-VITA

Five types of data were collected during the GT-VITA training process to evaluate the validity and utility of the tutor/aid paradigm as implemented in GT-VITA. The chapter first presents the general results and outcome of the training process. Second, the results and analysis of each data type are described. The five data types are (1) NASA personnel's evaluation of GT-VITA, (2) GT-VITA's performance assessments, (3) participants' knowledge questionnaires, (4) participants' performance evaluations, and (5) participants' evaluation of GT-VITA. The chapter ends with a discussion of the results.

General Results and Outcome of GT-VITA Training

All sixteen participants (thirteen trainees) were able to fulfill the entire training process in six to nine hours; all Group B subjects (i.e., the computer operators took nine hours). All lessons in the declarative phase were successfully completed by all trainees. During the procedural phase, the lessons of type *Learning Operations by Example* were successfully completed. Due to hardware limitations of TAE Plus, no trainee could complete lessons of type *Learning Operations by Doing*. In this event, the trainees were verbally briefed on the "missing pieces" and were given a written checklist of FOT operations to study (see Appendix D). Subsequently, the training process proceeded to the next phase.

With minimal assistance of the experimenter, every trainee was able to learn about and perform all FOT operations with the first lesson of the practice phase, *Practicing Operations with Feedback*. Although no trainee was prepared for this lesson, the lesson was so designed such that learning could occur through the reactive feedback points. Thereafter, all practice lessons were successfully completed.

NASA Personnel's Evaluation of GT-VITA

For three days after most of the participants had completed training, extended demonstrations of GT-VITA were arranged for NASA technical and training personnel in mission operations. Several weeks after the GT-VITA on-site training was completed, an in-house NASA preliminary evaluation of GT-VITA was conducted. Technical personnel who participated or saw the demonstrations, and training personnel who saw the demonstrations were asked to provide their subjective assessment of GT-VITA. Their comments are listed in a draft copy of the NASA evaluation report, as shown in Table 11-1. Table 11-1 also lists general comments about the potential long term benefits of using GT-VITA.

In general, the reactions from NASA personnel have been very good. The comments from the technical and training personnel were consistent with the goals and features of GT-VITA as an intelligent tutoring system. The perceived long term usefulness of GT-VITA was promising. The comments also indicate that more evaluation studies with larger sample size, for example, are imperative to test the long term benefits of GT-VITA. One NASA personnel had speculated that GT-VITA could reduce FOT training by as many as three months. Another personnel commented that even reducing training by one month could translate to major cost reduction for NASA (personal communications).

The positive results in this category are supported by both objective and subjective data collection during the GT-VITA training process. These data are discussed in the next four sections.

GT-VITA's Performance Assessments

During the practice phase of the GT-VITA training, performance assessment data were collected for every lesson scenario. The analysis of the data was performed separately for the two practice lessons.

Table 11-1. Comments on GT-VITA submitted by NASA Missions Operations Division

Technical Source Provided insight to flight operations that is not learned normally through regular computer operator and DOC controller training. Provides a better understanding of the support the computer operators and DOC controllers provide to the FOT. Brings a sense of importance to the computer operators and DOCs responsibilities and participation in mission operations. Provides a good understanding of the communications network interfaces and relationships which helps in the fault isolation and resolution process. Provides a good visualization of the end-to-end system interfaces and data flow of areas in the network unseen from the vantage point of the computer operators and DOCs. Taught cause and effect relationships that are normally experienced in real-time and not in current training.
Educator The training steps provide for excellent comprehension of information through example, visualization, and repetition. Each level added new information and the use of another skill little by little so as to not overload or discourage learning, but actually encouraged or excited learning form the student. Taught FOT responsibilities and way of thinking. Taught at a level for an FOT to learn their responsibilities and provide the level of detail needed to know as a spacecraft analyst, and was also easily comprehensible by persons not having the knowledge or responsibilities of an analyst.
Long Term Benefits Long term benefits have yet to be seen. The small sampling of computer operators and DOCs did not change how things are performed. Once everyone has used the tutor the changes will be more apparent. The ITS can be used as a yearly evaluation tool of performance for analysts, computer operators, and DOC controllers. Will also be used as initial training for analysts, computer operators, and DOC controllers.

Practicing Operations with Feedback

Table 11-2 summarizes the assessment data of all trainees for lessons of this type. A total of 35 lessons were completed. For each activity assessment, the number of times it occurred, and the number of times it was unsatisfactory due to missing actions were determined. Because not every activity was present in every lesson scenario, the percentage of assessment error for each activity was computed for comparison. For activities that involved more than one operator actions, a decomposition of the assessment errors attributed to the different actions was determined as shown in Table 11-3.

Table 11-2. Summary of GT-VITA's assessment data for lesson type *Practicing Operations with Feedback.*

Total Number of Scenarios Completed = 35				
	Assessment Type	**Number of Occurrences**	**Number of Errors**	**Percentage of Errors**
Pre-Pass:	CONFIG	35	4	11
	VERIFY	35	32	91
	DCI	35	13	37
	BOOK	35	10	29
On-Pass:	FAIL-MGT	31	15	48
	FAIL-BOOK	31	22	71
	ANOM-BOOK	31	19	61
	MONITOR	34	23	68
	BOOK	34	22	65
Post-Pass:	TERM	34	9	26
	BOOK	34	10	29

Results show that participants had the most trouble with verification during the pre-pass phase, mainly due to participants' missing actions to check support schedule, Application Processor's software and Command Storage Memory listing. Being the first pre-pass activity undertaken, configuration of the Application Processor and the Telemetry and Command computer was done with least errors. Because the duration of a pass was much longer than the pre-pass and post-pass phases combined, there were more opportunities for errors as reflected by the relatively higher percentages of assessment errors durint a pass.

Table 11-3. Breakdown of assessment errors for lesson type *Practicing Operations with Feedback.*

	Assessment Error Type	Missing Actions	Frequency of Occurrence
Pre-Pass:	VERIFY		32
		Check support schedule	31
		Check AP software	26
		Check CSM listing	31
		Request ODMs	10
	CONFIG		4
		Initialize TAC	4
		Initialize AP	1
On-Pass:	MONITOR		23
		Display events page	17
		Execute TDRSSCHK	12
		Execute ERBECHK	13
		Execute SAGECHK	13
		Display CDHS page	2
		Display ACS page	1
		Display RFS page	1
		Display PS page	1
		Display TAC page	1

A decomposition of the monitoring functions shows that participants most often missed the action to display the events page, followed by actions to perform some STOL monitoring procedures.

For this type of practice lessons, participants were learning to coordinate the various FOT operations and to perform them in a timely manner. Thus, an assessment error, which translated to one or more missing actions, could be committed for three reasons. First, a participant forgot about the required actions. For instance, during the pass, participants often forgot to execute the STOL procedures because they were busy monitoring the NASA graphic displays. Second, the actions were not done on time as expected by the tutor. For instance, the tutor allowed only two minutes to complete all pre-pass activities. So, during the first few practice scenarios, participants often complained about the lack of time to complete the verification function. Other times, some of the verification actions were plain forgotten. Third, the actions were not perceived as necessary. For instance, while a participant was busy attending to a ground equipment failure,

a spacecraft parameter value became abnormal, but was never detected. Thus, no action was taken for the problem. At this time, GT-VITA does not differentiate between the three types of missing actions.

To find out if there were any group differences in the assessment data, the percentages of assessment errors were re-computed for each subject group, as shown in Table 11-4. Two nonparametric statistical procedures were performed. First, the Friedman's two-way analysis of variance tested the null hypothesis that the assessment errors were committed equally among the subject groups. The calculations are shown in Table 11-5. The results show that the null hypothesis could not be rejected at a significance level of 0.05; that is, there were no group difference in percentage of assessment errors committed.

Second, a complementary analysis computed the Kendall's coefficient of concordance to test the null hypothesis that when the assessment error percentages were ranked for each group, there was no agreement between the rankings. The calculations are shown in Table 11-6. The results show that the null hypothesis could not be accepted at a significance level of 0.05; that is, rankings of assessment percentages were in agreement between groups.

Table 11-4. Assessment error percentages by subject group for lesson type *Practicing Operation with Feedback.*

		Subject Group			
	Assessment Type	**A**	**B**	**C**	**D**
Pre-Pass:	CONFIG	17	0	22	0
	VERIFY	83	100	100	86
	DCI	17	29	67	43
	BOOK	0	57	56	14
On-Pass:	FAIL-MGT	73	57	29	17
	FAIL-BOOK	82	100	57	33
	ANOM-BOOK	60	100	38	50
	MONITOR	50	71	63	100
	BOOK	58	71	75	57
Post-Pass:	TERM	25	14	38	29
	BOOK	17	43	25	43
Number of Scenarios Completed		12	7	9	7

Table 11-5. Freidman's two-way analysis of variance for lesson type *Practicing Operation with Feedback.* Cell numbers are ranks of assessment error percentages by rows.

	Subject Group (treatment, n = 4)			
Assessment Type (blocks, k = 11)	**A**	**B**	**C**	**D**
CONFIG	3	1.5	4	1.5
VERIFY	1	3.5	3.5	2
DCI	1	2	4	3
PRE-BOOK	1	4	3	2
FAIL-MGT	4	3	2	1
FAIL-BOOK	3	4	2	1
ANOM-BOOK	3	4	1	2
MONITOR	1	3	2	4
ON-BOOK	2	3	4	1
TERM	2	1	4	3
POST-BOOK	1	3.5	2	3.5
Sum of Ranks, ΣR	22	32.5	24	31.5

H_0: The average percentage of assessment errors is the same irrespective of subject group.

$S = \Sigma R^2 - k^2 n(n+1)/4 = 83.5$
$F = 12S/kn(n+1) = 4.555 < \text{Chi Square}_{(0.05,\ 3)} = 7.814$; cannot reject H_0.

Table 11-6. Kendall's coefficient of concordance for lesson type *Practicing Operation with Feedback.* Cell numbers are ranks of assessment error percentages by columns.

	Subject Group (sets, n = 4)				
Assessment Type (objects, n= 11)	**A**	**B**	**C**	**D**	**Sum of Ranks, ΣR**
CONFIG	3	1	1	1	6
VERIFY	11	10	11	10	42
DCI	3	3	9	6.5	21.5
PRE-BOOK	1	5.5	6	2	14.5
FAIL-MGT	9	5.5	3	3	20.5
FAIL-BOOK	10	10	7	5	32
ANOM-BOOK	8	10	4.5	8	30.5
MONITOR	6	7.5	8	11	32.5
ON-BOOK	7	7.5	10	9	33.5
TERM	5	2	4.5	4	15.5
POST-BOOK	3	4	2	6.5	15.5

H_0: There is no agreement between the four subject group rankings of assessment error percentages

$S = \Sigma R^2 - k^2 n(n+1)/4 = 1170$
Coefficient of concordance, $W = 12S/k^2 n(n^2-1) = 0.665$
$F = k(n-1)W = 26.591 > \text{Chi Square}_{(0.05,\ 10)} = 18.307$; cannot accept H_0.

Practicing Operations with Checkpoints

Table 11-7 summarizes the assessment data of all trainees for lessons of this type. A total of 34 lessons were completed. The percentage of assessment errors for each activity at the checkpoints was computed for comparison. A decomposition of the VERIFY and MONITOR assessment errors is shown in Table 11-8.

In these lessons, participants were practicing coordination and performance efficiency of GT-POCC operations on their own. The tutor only intervened at checkpoints to alert about assessment errors. Results show that participants were more often stopped because of missing verification actions at the pre-pass checkpoint, monitoring actions at the on-pass checkpoint, and bookkeeping actions at both on-pass and post-pass checkpoints. Participants were successful in completing the configuration, ground failure management and termination functions in all lesson scenario.

Notice that the verification and monitoring functions had a similar relationship of assessment errors as the previous practice lesson type. Even when given sufficient time, participants sometimes still seemed to forget to perform the verification checks, the STOL procedures, and events display action. The visibility of these and other "forgotten actions" on the GT-VITA interface might be the root of the problem, a point that will be addressed in another section.

Figure 11-1 shows the distribution of the different checkpoint errors committed by participants. Out of the 34 lesson scenarios, four were completed without any tutorial interventions. Since the on-pass phase involved the most activities, it was not surprising that on-pass checkpoint errors were committed the most in all lessons, either in combination with pre-pass checkpoint errors, or as the only error type.

Statistical analysis similar to the previous practice lesson type was performed for the assessment data to test for group differences. The assessment data by subject group are shown in Table 11-9. The calculations for Friedman's two-way analysis of variance and Kendall's coefficient of concordance are presented in Tables 11-10 and 11-11. The results were similar to the previous analysis also. Specifically, the assessment errors were committed equally among the subject groups, and the rankings of error percentages were in agreement between the subject groups.

Table 11-7. Summary of GT-VITA's assessment data for lesson type *Practicing Operations with Checkpoints.*

Total Number of Scenarios Completed = 34				
	Assessment Type	**Number of Occurrences**	**Number of Errors**	**Percentage of Errors**
At Pre-Pass Checkpoint:	CONFIG	34	0	0
	VERIFY	34	13	38
	DCI	34	1	3
	BOOK	34	3	9
At On-Pass Checkpoint:	FAIL-MGT	28	0	0
	FAIL-BOOK	28	4	14
	ANOM-BOOK	30	3	10
	MONITOR	34	9	26
	BOOK	34	11	32
At On-Pass Checkpoint:	TERM	34	0	0
	BOOK	34	7	20

Table 11-8. Breakdown of assessment errors for lesson type *Practicing Operations with Checkpoints.*

	Assessment Error Type	**Missing Actions**	**Frequency of Occurrence**
At Pre-Pass Checkpoint:	VERIFY		13
		Check support schedule	7
		Check AP software	5
		Check CSM listing	11
		Request ODMs	2
At On-Pass Checkpoint:	MONITOR		9
		Display events page	5
		Execute TDRSSCHK	2
		Execute ERBECHK	2
		Execute SAGECHK	2
		Display TAC page	2
		Display RFS page	1

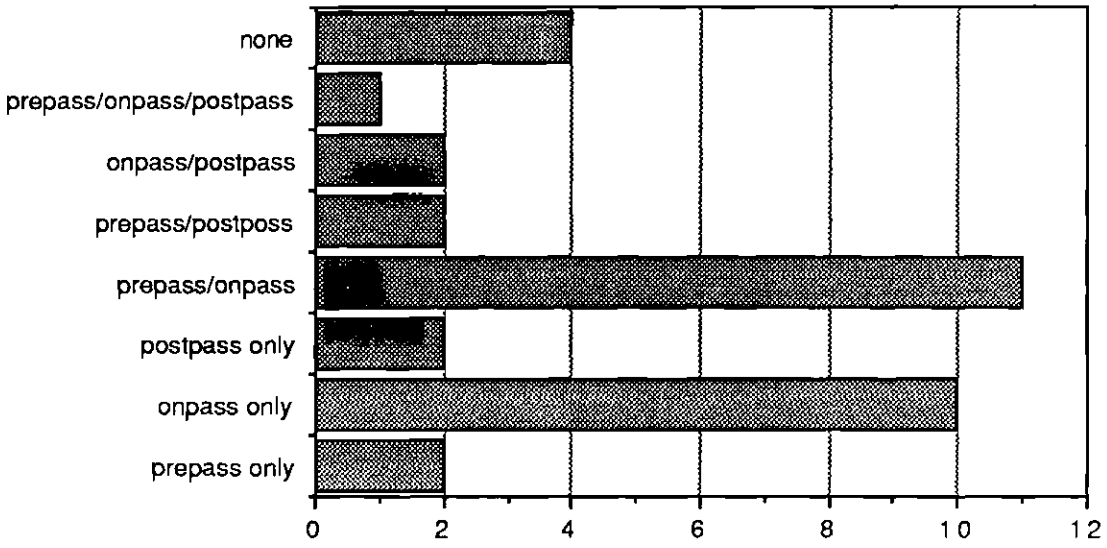


Figure 11-1. Distribution of different checkpoint errors committed

Table 11-9. Assessment error percentages by subject group for lesson type *Practicing Operation with Checkpoints.*

		Subject Group			
	Assessment Type	**A**	**B**	**C**	**D**
At Pre-Pass Checkpoint:	CONFIG	0	0	0	0
	VERIFY	40	0	71	33
	DCI	0	0	0	8
	BOOK	10	0	14	8
At On-Pass Checkpoint:	FAIL-MGT	0	0	0	0
	FAIL-BOOK	33	0	0	10
	ANOM-BOOK	13	25	0	9
	MONITOR	30	40	29	17
	BOOK	30	60	29	25
At On-Pass Checkpoint:	TERM	0	0	0	0
	BOOK	10	40	29	17
Number of Scenarios Completed		10	5	7	12

Table 11-10. Freidman's two-way analysis of variance for lesson type *Practicing Operation with Checkpoints*. Cell numbers are ranks of assessment error percentages by rows.

	Subject Group (treatment, n = 4)			
Assessment Type (blocks, k = 11)	**A**	**B**	**C**	**D**
CONFIG	2.5	2.5	2.5	2.5
VERIFY	4	1	3	2
DCI	2	2	2	4
PRE-BOOK	3	1	4	2
FAIL-MGT	2.5	2.5	2.5	2.5
FAIL-BOOK	4	1.5	1.5	3
ANOM-BOOK	3	4	1	2
MONITOR	3	4	2	1
ON-BOOK	3	4	2	1
TERM	2.5	2.5	2.5	2.5
POST-BOOK	1	4	3	2
Sum of Ranks, ΣR	30.5	29	26	24.5

H_0: The average percentage of assessment errors is the same irrespective of subject group.

$S = \Sigma R^2 - k^2 n(n+1)/4 = 22.5$

$F = 12S/kn(n+1) = 1.227 < \text{Chi Square}_{(0.05,\ 3)} = 7.814$; cannot reject H_0.

Table 11-11. Kendall's coefficient of concordance for lesson type *Practicing Operation with Checkpoints*. Cell numbers are ranks of assessment error percentages by columns.

	Subject Group (sets, n = 4)				
Assessment Type (objects, n= 11)	**A**	**B**	**C**	**D**	**Sum of Ranks, ΣR**
CONFIG	2.5	4	3.5	2	12
VERIFY	11	4	11	11	37
DCI	2.5	4	3.5	4.5	14.5
PRE-BOOK	5.5	4	7	4.5	21
FAIL-MGT	2.5	4	3.5	2	12
FAIL-BOOK	10	4	3.5	7	24.5
ANOM-BOOK	7	8	3.5	6	24.5
MONITOR	8.5	9.5	9	8.5	3 .5
ON-BOOK	8.5	11	9	10	38.5
TERM	2.5	4	3.5	2	12
POST-BOOK	5.5	9.5	9	8.5	32.5

H_0: There is no agreement between the four subject group rankings of assessment error percentages

$S = \Sigma R^2 - k^2 n(n+1)/4 = 1115.5$

Coefficient of concordance, $W = 12S/k^2 n(n^2-1) = 0.634$

$F = k(n-1)W = 25.35 > \text{Chi Square}_{(0.05,\ 10)} = 18.307$; cannot accept H_0.

In short, in both practice lesson types, the statistical results suggest that each subject group faired the same in the percentage of each assessment error committed, and that activities ranked the same among the groups in terms of how "manageable" they were. As described in Chapter X, the subject groups consisted of NASA government personnel, computer operators, spacecraft specialists and software testing team. The objective performance results reported here suggest that GT-VITA provides a practice environment for FOT operations that is sufficiently robust to accommodate subjects of varying background in knowledge and experience, and in computer literacy.

Participants' Knowledge Questionnaires

Posttest questionnaires were given verbally to the participants after declarative and procedural phases of training. In general, all participants were able to answer most questions about system objects and relations, and GT-POCC operations. Some participants sought answers right away from the GT-VITA interfaces made accessible to them. Others attempted to answer the questions without looking at the GT-VITA interfaces until they came to an impasse.

Participants' Performance Evaluations

Figure 11-2 shows results of the questionnaire conducted after the declarative training. Overall, participants felt they understood almost all GT-POCC objects and relations. Only a few trainees felt "neutral" towards what they knew: two about MSOCC, four about software failures, and one about support configuration. A neutral response could mean thatg participants understood some but not all aspects of a declarative element, or that they did not or missed the chance to learn about it. The former explanation would apply to MSOCC and support configuration which had many aspects to each element. The latter explanation would apply to software failures. Although explanation for software failures was available, a participant might not see the software failure demonstrated in real time, either because the event was not included in a scenario, or the participant was busy with other system dynamics.

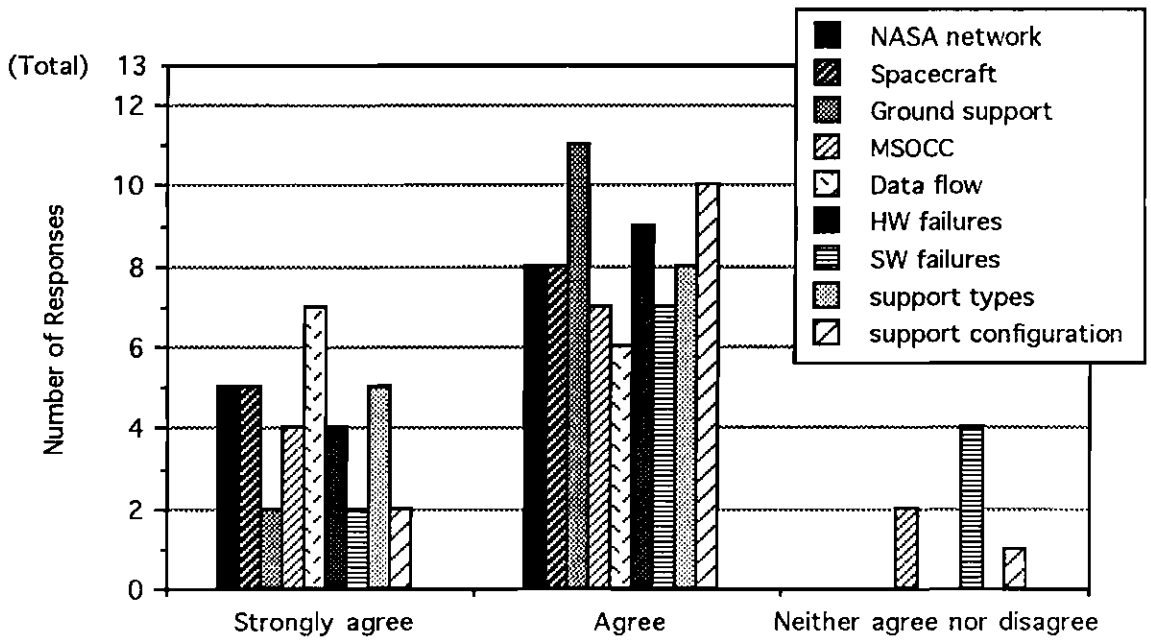


Figure 11-2. Students' perception of their understanding of GT-POCC objects and relations.

Figures 11-3(a), (b) and (c) show results of the questionnaire conducted after the procedural training. All but one participants felt they understood GT-POCC operations in terms of pass procedures, trouble management and spacecraft commanding. One participant felt neutral towards what he knew about spacecraft commanding. It could be perceived that relative to pass procedures and trouble management, spacecraft commanding operations were less understood because of their complexity.

Figures 11-4(a), (b) and (c) show results of the performance portion in the final questionnaire. At the end of the training, all participants felt confident about performing pass procedures and trouble management operations. Interestingly, even though no one had practice with spacecraft commanding, all but one participants were confident about performing commanding activities. One participant felt neutral, the same one who felt neutral towards spacecraft commanding in the previous questionnaire. A note about the total of responses for these questionnaires is in order here. Chapter VIII reported that six out of the 16 participants were trained in pairs. The first two questionnaires were given to the 13 *trainees*, while the final questionnaire was given to all 16 subjects.

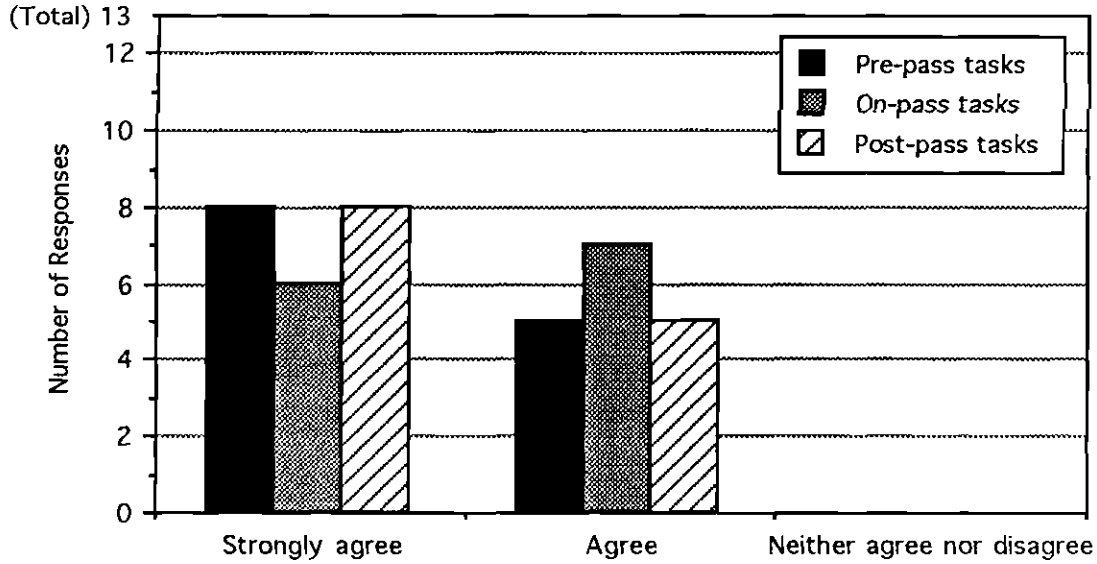


Figure 11-3(a). Students' perception of their understanding of GT-POCC operations (pass procedures).

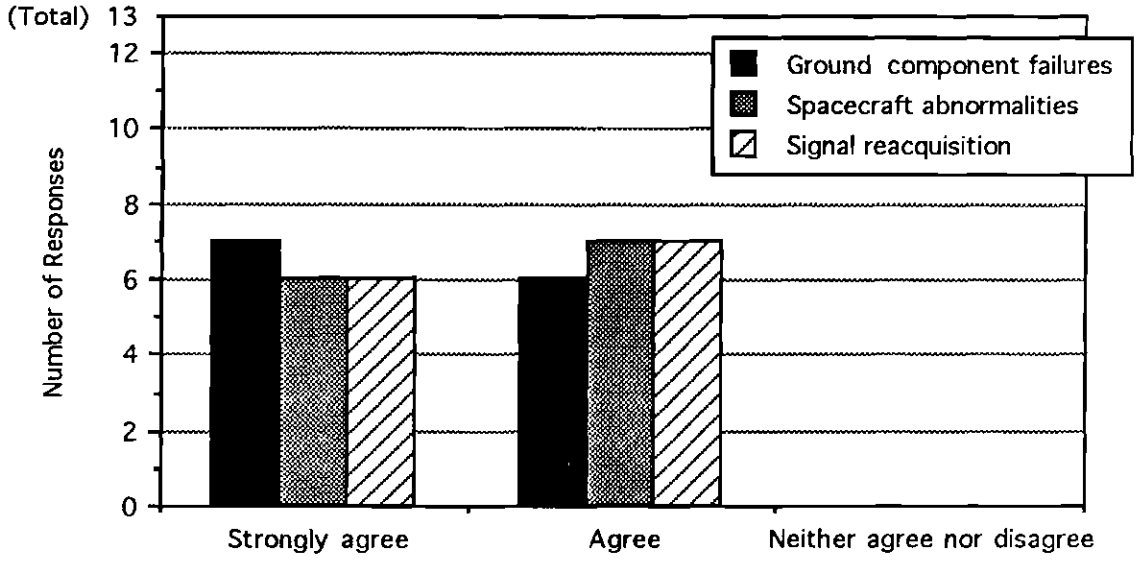


Figure 11-3(b). Students' perception of their understanding of GT-POCC operations (trouble management).

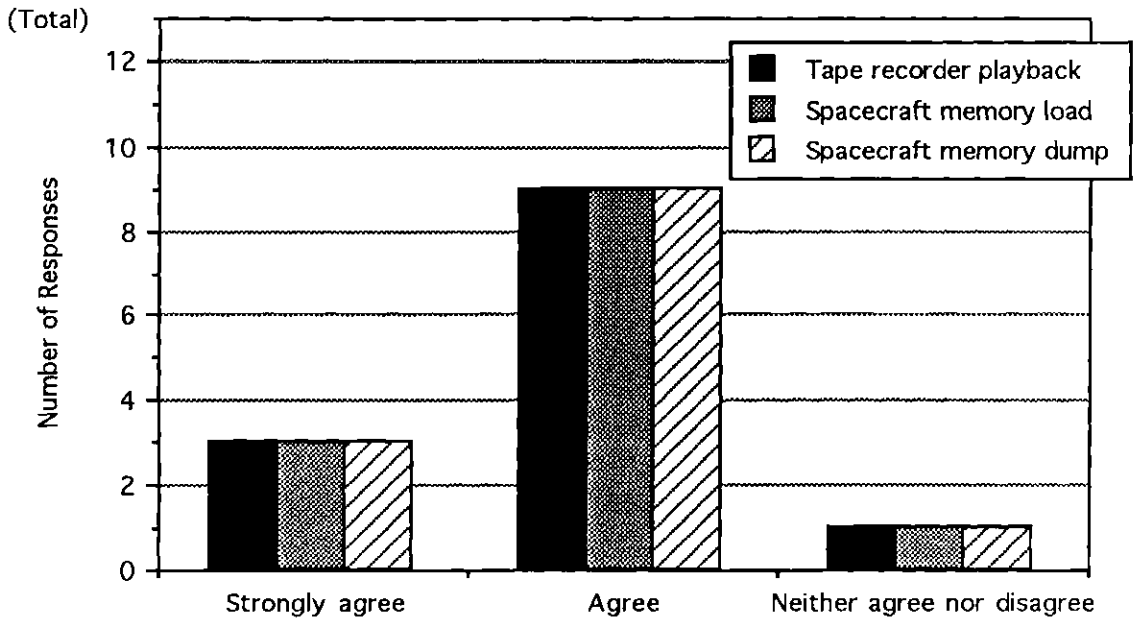


Figure 11-3(c). Students' perception of their understanding of GT-POCC operations (spacecraft commanding).

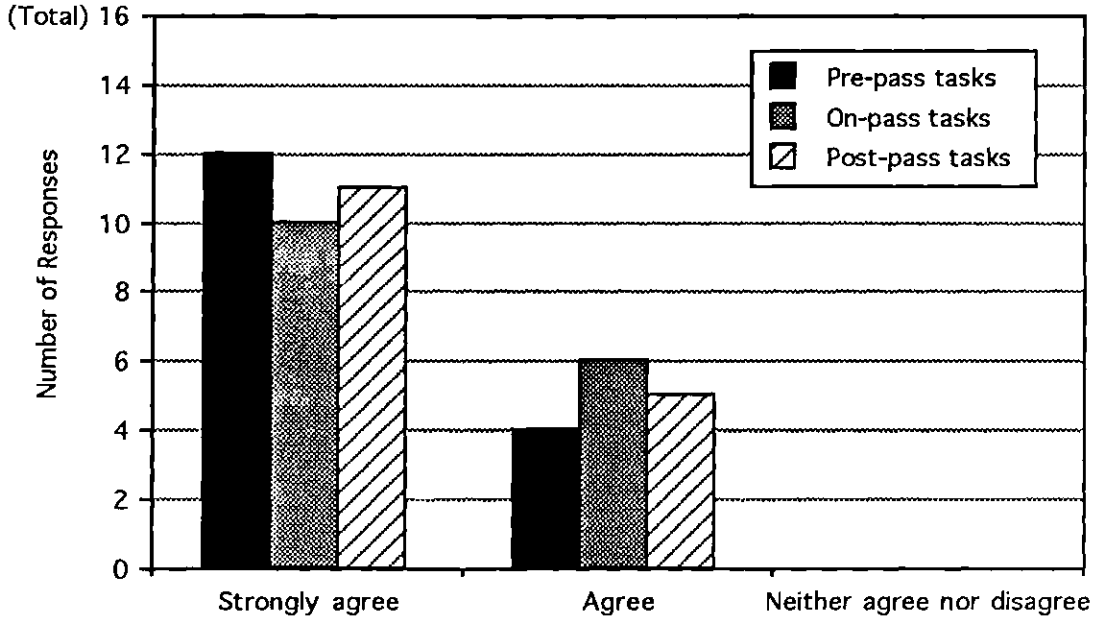


Figure 11-4(a). Students' perception of their confidence in performing GT-POCC operations (pass procedures).

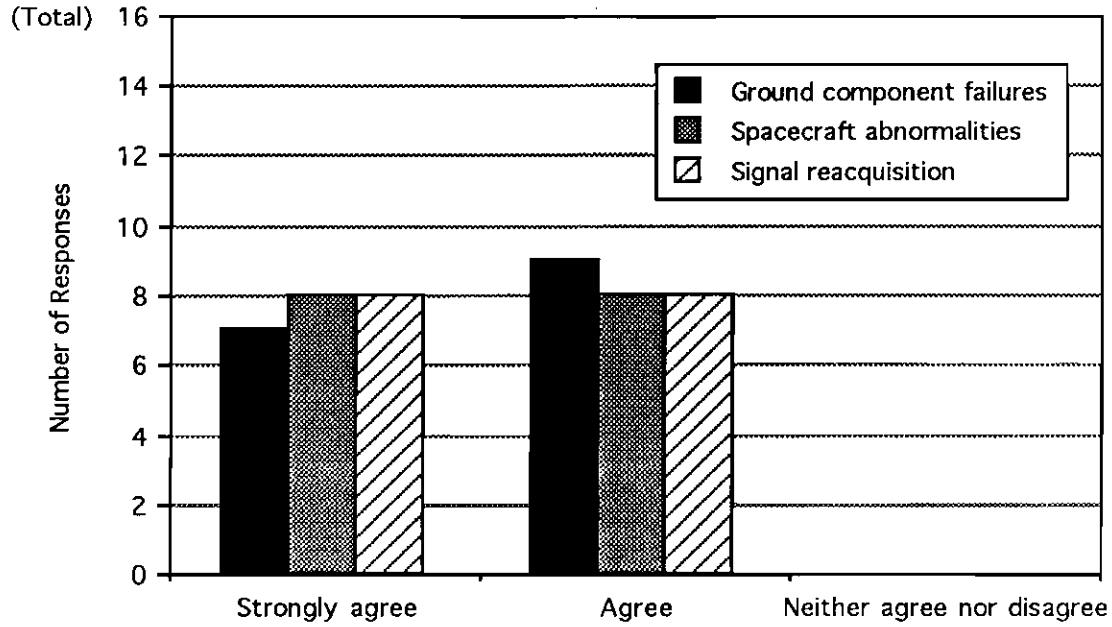


Figure 11-4(b). Students' perception of their confidence in performing GT-POCC operations (trouble management).

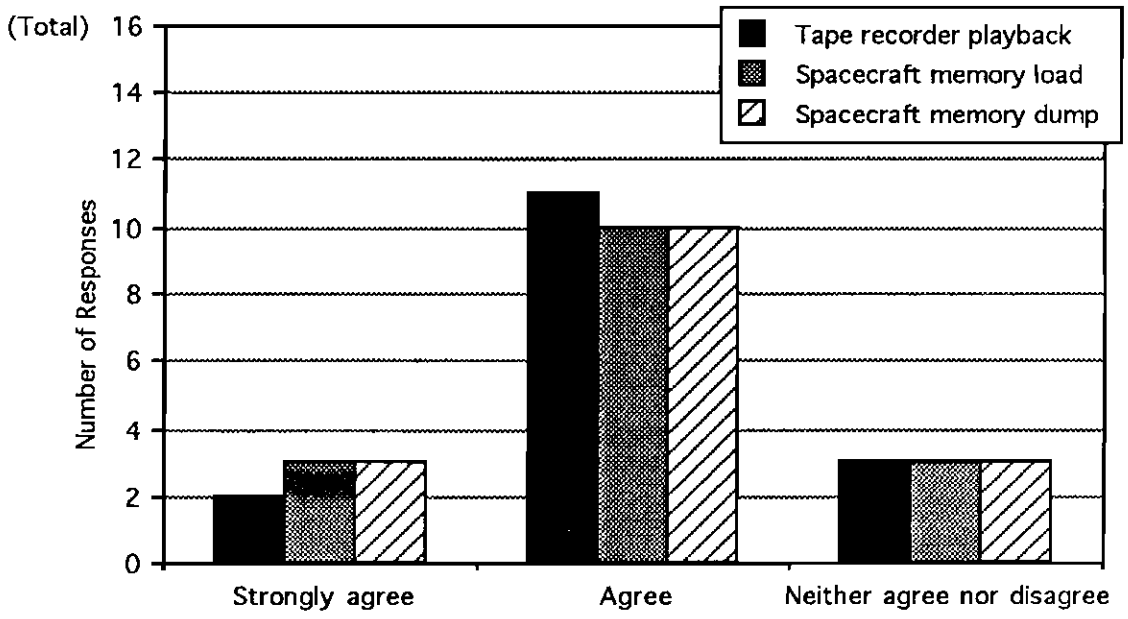


Figure 11-4(c). Students' perception of their confidence in performing GT-POCC operations (spacecraft commanding).

In short, the results were very favorable; participants felt they learn the knowledge and skills of an FOT analyst. These results suggest that subjects were comfortable and confident with the learning environment and instruction provided by GT-VITA in acquiring the declarative and procedural knowledge demanded of FOT analysts. The graphical visualization of the NASA system is a signicifant improvement over the many volumes of manuals and low level task commands that are available to the student to learn. This point was well supported by an experienced COBE FOT analyst passing by the experimental setup who commented how she wished GT-VITA was in existence when she was being trained.

Participants' Subjective Evaluation of GT-VITA

At the end of training, all sixteen participants were asked to evaluate GT-VITA in terms of its overall usefulness and the effectiveness of the lesson types administered. Figures 11-5, 11-6 and 11-7 capture participants' reaction to GT-VITA. Overall, Figure 11-5 shows that all participants agree (with thirteen of sixteen strongly agree) that GT-VITA is a useful tool for tutoring new FOT analysts.

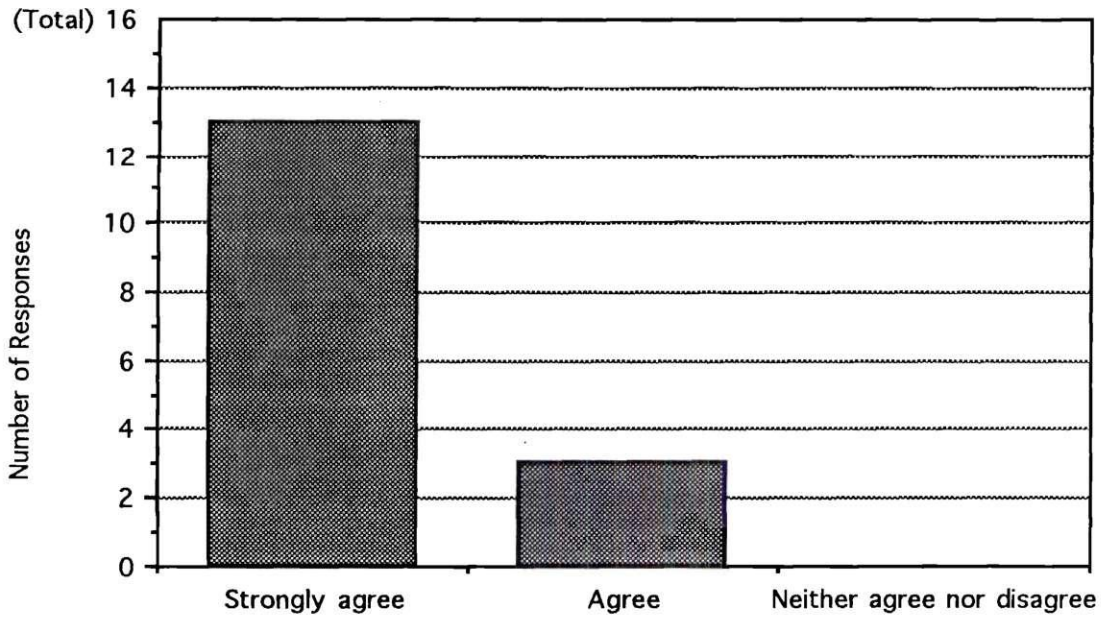


Figure 11-5. Students' perception of GT-VITA as a useful tool for tutoring new FOT analysts

In terms of lesson types, Figure 11-6(a) shows that the declarative lessons types contributed to participants' knowledge of GT-POCC with a few exceptions. On a scale from 1 to 10, with 10 being the best, five responses scored below 5. Three of the responses came from the same subject (A1) rating negatively for all declarative lesson types. Subjects A1 and D4 gave the worst score to the lesson *Exploring Tutor's Knowledge*. Subject A3 also rated the same lesson unfavorably. The slightly negative response to the lesson type *Exploring Tutor's Knowledge* might be due to the minimal time participants could only spend on it and also due to the fact that without the objective for transition from tutor to aid, this lesson seemed out of context at the time of training. Figure 11-6(b) shows that all procedural and practice lesson types contributed favorably to participant's knowledge of GT-POCC; all lesson types scored 7 or higher.

Participants were asked to rank the lesson types again to assess their potential usefulness if GT-VITA is fully implemented. Figure 11-7(a) shows that all declarative lesson types were considered very useful with one exception. Subjects A1 and D4 rated all lesson types positively; even though some lesson types did not contributed much to their knowledge, they would still be potentially useful in actual training. Only subject A3 still rated the lesson type *Exploring Tutor's Knowledge* unfavorably. Figure 11-7(b) shows that all procedural and practice lesson types were considered very useful; all lesson types scored 7 or above.

The average ranking for each lesson type were computed for both sets of data on contribution and usefulness, as shown in Tables 11-12 and 11-13. The tables also show the average rankings computed for each subject group. Statistical analysis was performed on both sets of data. Table 11-14 shows the calculations for the Freidman's two-way analysis of variance on the contribution data. Significant difference was found in how each group rated the lesson types. The difference is also evident by visual inspection of the data. The striking result is that Group B, the computer operators, consistently rated all lesson types except one the highest. Moreover, the collapsed average score for all lesson types was highest for Group B, and much higher than other groups. Table 11-15 shows the calculations for Kendall's coefficient of concordance. Significant agreement was found between subject groups in how they rank the lesson types'

contribution to their GT-POCC knowledge. Visually, it is evident that the procedural and practice lesson types were consistently scored higher than the declarative ones, regardless of subject group.

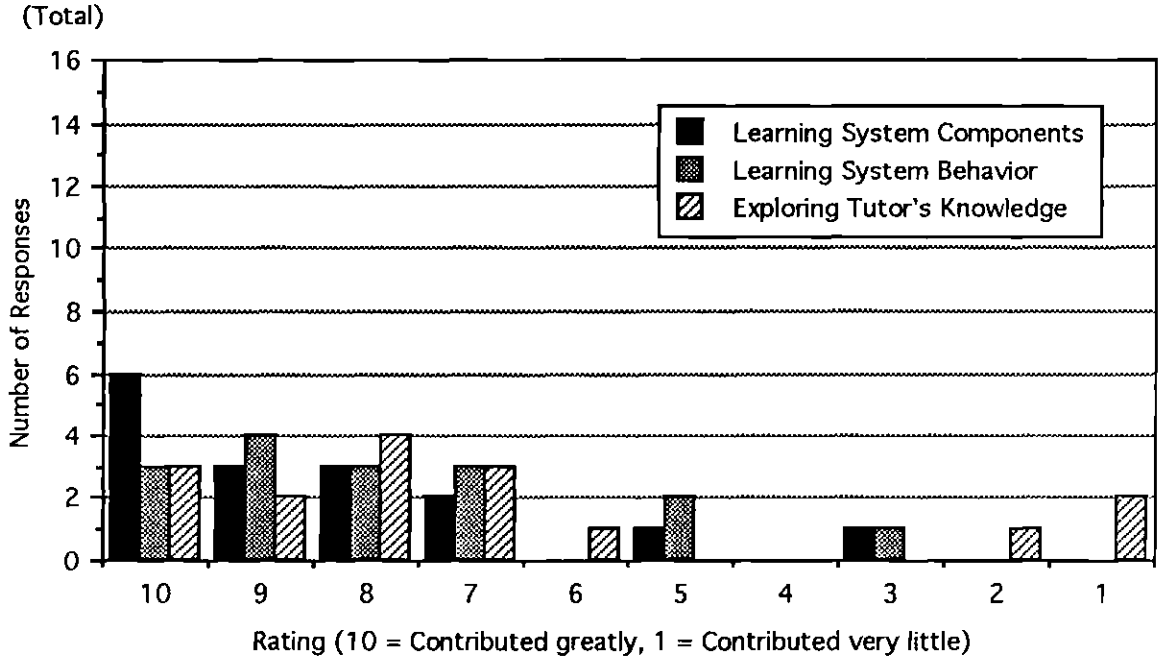


Figure 11-6(a). Students' rating of each lesson type's contribution to their overall knowledge (declarative lessons)

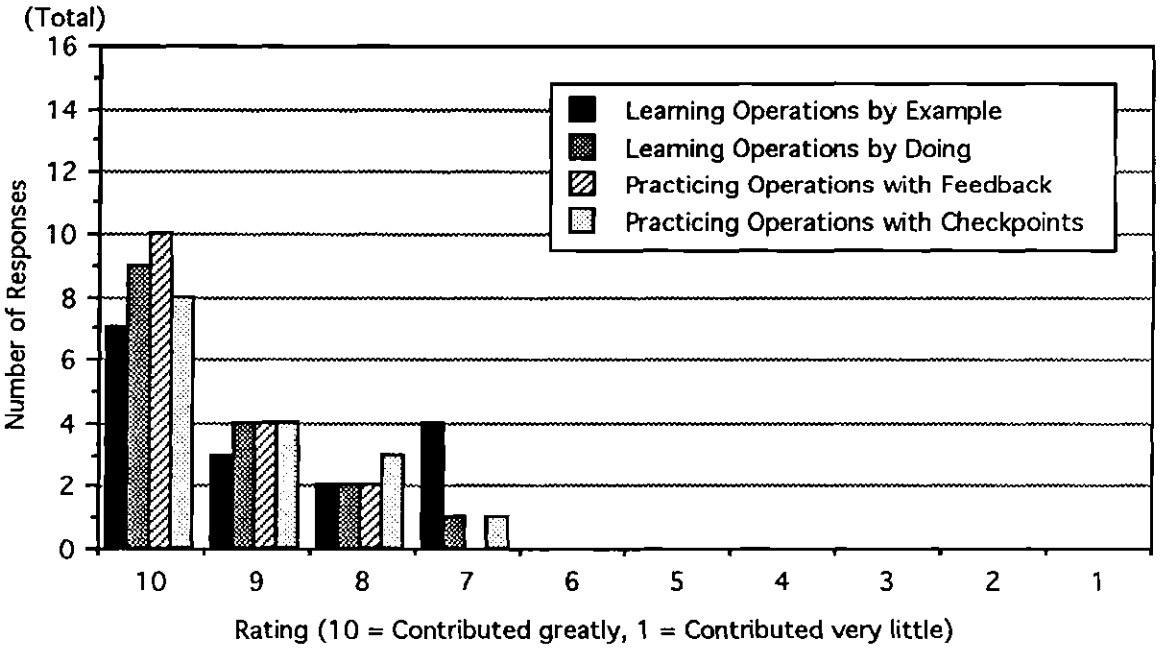


Figure 11-6(b). Students' rating of each lesson type's contribution to their overall knowledge (procedural and practice lessons)

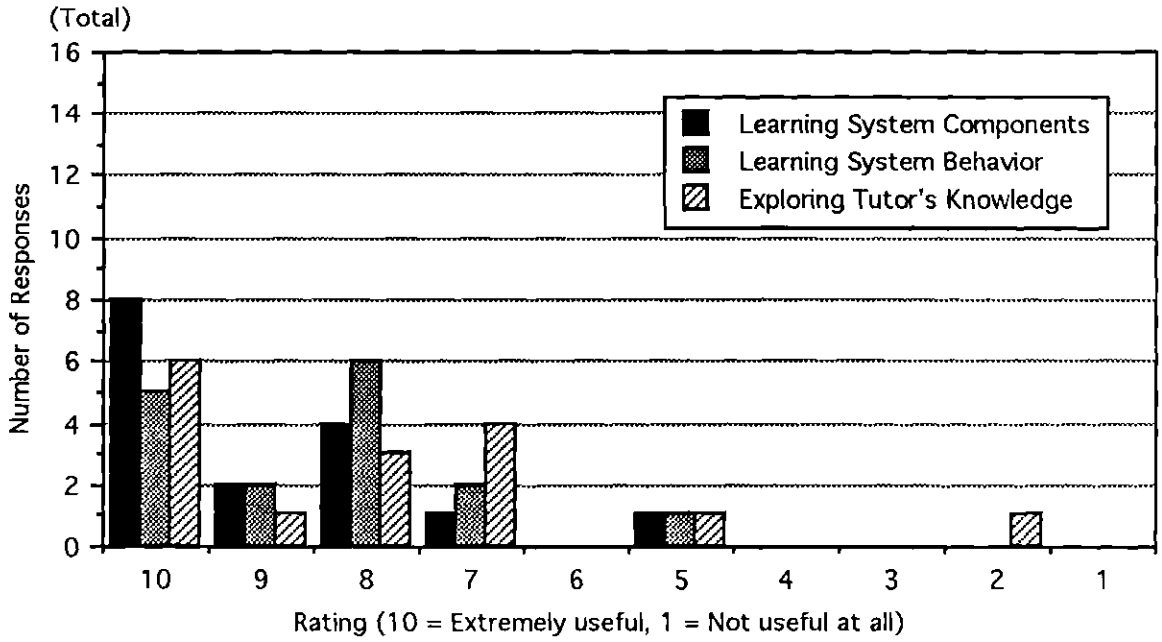


Figure 11-7(a). Students' rating of each lesson type's potential usefulness if GT-VITA is fully implemented (declarative lessons)

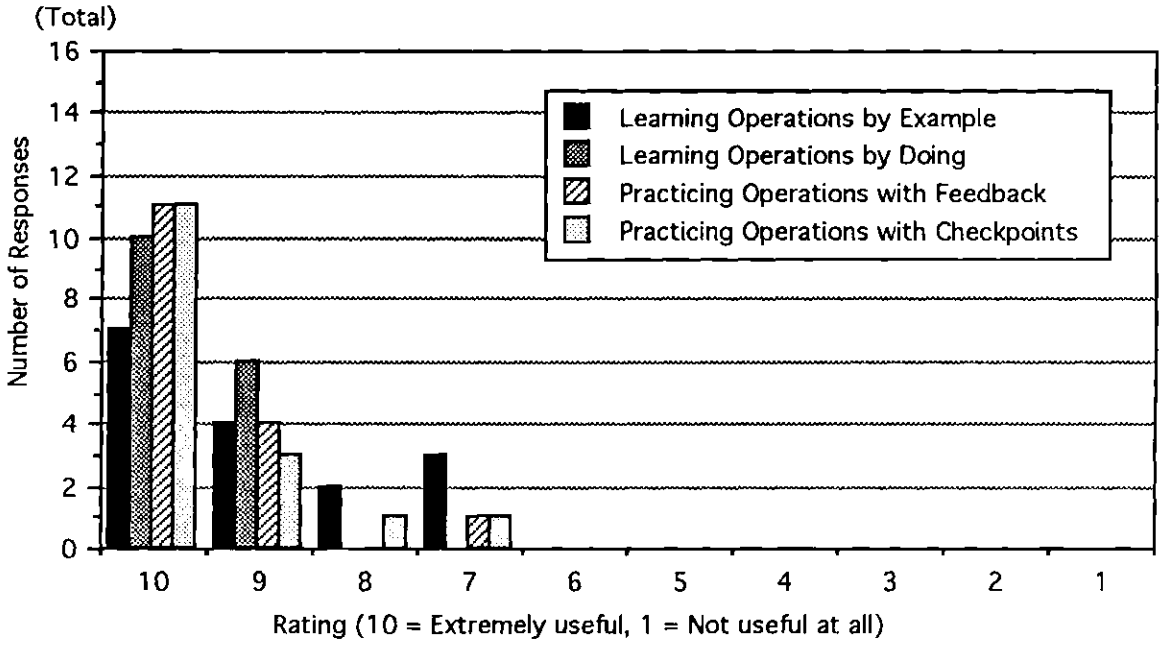


Figure 11-7(b). Students' rating of each lesson type's potential usefulness if GT-VITA is fully implemented (procedural and practice lessons)

Table 11-12. Average rankings of each lesson type's contribution to subjects' overall knowledge of GT-POCC.

	Subject Group				
Lesson Type	**A**	**B**	**C**	**D**	**Mean**
Practicing Operations with Checkpoints	6.75	10	8	8.67	8.35
Learning Operations by Example	6.75	9.33	7.67	7.67	7.85
Learning Operations by Doing	5.5	9.67	7	6.5	7.17
Learning System Components	9	10	8	8.5	8.87
Learning System Behavior	9.25	10	8.33	9.5	9.27
Practicing Operations with Feedback	9.75	9.67	9.33	9.33	9.52
Exploring Tutor's Knowledge	9.25	9.67	8.33	9.33	9.15
Mean	8.03	9.76	8.1	8.5	8.60

Table 11-13. Average rankings of each lesson type's potential usefulness if GT-VITA is fully implemented.

	Subject Group				
Lesson Type	**A**	**B**	**C**	**D**	**Mean**
Practicing Operations with Checkpoints	8.5	10	8.67	8.67	8.96
Learning Operations by Example	8.5	10	7.67	8	8.54
Learning Operations by Doing	7.75	10	7	7.67	8.10
Learning System Components	9	10	8	8.5	8.87
Learning System Behavior	9.25	10	9.33	9.67	9.56
Practicing Operations with Feedback	9.75	10	9.67	9.17	9.65
Exploring Tutor's Knowledge	9	10	9	9.33	9.33
Mean	8.82	10	8.48	8.71	9.00

Similar statistical analysis was conducted on the usefulness data. Table 11-16 shows the calculations for Friedman's two-way analysis of variance. Significant difference was found in how each group rated the lesson types' potential usefulness. Group B again gave the highest score to all lesson types, which was strikingly higher that those of other groups. Table 11-17 shows the calculations for Kendall's coefficient of concordance. Significant agreement in the rankings of lesson types between groups was achieved at an alpha level of 0.10.

In summary, the results in this category show that participants reacted very positively to GT-VITA as a useful training tool. Furthermore, participants reacted very favorably to GT-VITA's pedagogical approach which was rated highest by NASA computer operators but appealed generally to all subject groups in the same way. The strikingly positive response from the computer operators is not surprising, but rather consistent with the nature of the subject group. Relative to the rest of the subject groups, the computer

Table 11-14. Freidman's two-way analysis of variance for lesson types' contribution to subjects' overall knowledge of GT-POCC. Cell numbers are ranks of lesson types by rows.

	Subject Group (treatments, n=4)			
Lesson Type (blocks, k=7)	**A**	**B**	**C**	**D**
Practicing Operations with Checkpoints	4	1	3	2
Learning Operations by Example	4	1	2.5	2.5
Learning Operations by Doing	4	1	2	3
Learning System Components	2	1	4	3
Learning System Behavior	3	1	4	2
Practicing Operations with Feedback	1	2	3.5	3.5
Exploring Tutor's Knowledge	3	1	4	2
Sum of Ranks, ΣR	21	8	23	18

H_0: The average ranking of lesson types is the same irrespective of subject group.

$S = \Sigma R^2 - k^2 n(n+1)/4 = 133$

$F = 12S/kn(n+1) = 11.4 >$ Chi Square$_{(0.05, 3)} = 7.814$; cannot accept H_0.

Table 11-15. Kendall's coefficient of concordance for lesson types' contribution to subjects' overall knowledge of GT-POCC. Cell numbers are ranks of lesson types by columns.

	Subject Group (sets, k = 4)				
Lesson Type (objects, n=7)	**A**	**B**	**C**	**D**	**ΣR**
Practicing Operations with Checkpoints	5.5	2	4.5	4	16
Learning Operations by Example	5.5	7	6	6	24.5
Learning Operations by Doing	7	5	7	7	26
Learning System Components	4	2	4.5	5	15.5
Learning System Behavior	2.5	2	2.5	1	8
Practicing Operations with Feedback	1	5	1	2.5	9.5
Exploring Tutor's Knowledge	2.5	5	2.5	2.5	12.5

H_0: There is no agreement between the four subject group rankings of lesson types

$S = \Sigma R^2 - k^2 n(n+1)/4 = 291$

Coefficient of concordance, $W = 12S/k^2 n(n^2-1) = 0.650$

$F = k(n-1)W = 15.589 >$ Chi Square$_{(0.05, 6)} = 12.592$; cannot accept H_0.

Table 11-16. Freidman's two-way analysis of variance for lesson types' potential usefulness if GT-VITA is fully implemented. Cell numbers are ranks of lesson types by rows.

	Subject Group (treatments, n=4)			
Lesson Type (blocks, k=7)	**A**	**B**	**C**	**D**
Practicing Operations with Checkpoints	4	1	2.5	2.5
Learning Operations by Example	3	1	4	2
Learning Operations by Doing	2	1	4	3
Learning System Components	2	1	4	3
Learning System Behavior	4	1	3	2
Practicing Operations with Feedback	2	1	3	4
Exploring Tutor's Knowledge	3.5	1	3.5	2
Sum of Ranks, ΣR	20.5	7	24	18.5

H_0: The average ranking of lesson types is the same irrespective of subject group.

$S = \Sigma R^2 - k^2 n(n+1)/4 = 162.5$

$F = 12S/kn(n+1) = 13.929 >$ Chi Square$(0.05, 3) = 7.814$; cannot accept H_0.

Table 11-17. Kendall's coefficient of concordance for lesson types' potential usefulness if GT-VITA is fully implemented. Cell numbers are ranks of lesson types by columns.

	Subject Group (sets, k = 4)				
Lesson Type (objects, n=7)	**A**	**B**	**C**	**D**	**ΣR**
Practicing Operations with Checkpoints	5.5	4	4	4	17.5
Learning Operations by Example	5.5	4	6	6	21.5
Learning Operations by Doing	7	4	7	7	25
Learning System Components	3.5	4	5	5	17.5
Learning System Behavior	2	4	2	1	9
Practicing Operations with Feedback	1	4	1	3	9
Exploring Tutor's Knowledge	3.5	4	3	2	12.5

H_0: There is no agreement between the four subject group rankings of lesson types

$S = \Sigma R^2 - k^2 n(n+1)/4 = 226$

Coefficient of concordance, $W = 12S/k^2 n(n^2-1) = 0.504$

$F = k(n-1)W = 12.107 <$ Chi Square$(0.05, 6) = 12.59$; cannot reject H_0, at 0.05 significance

$>$ Chi Square$(0.10, 6) = 10.6$; cannot accept H_0, at 0.10 significance

operators had the least background knowledge and computer experience. Thus, they had the most to gain from the GT-VITA training which was perceived to affect their future job advancements. Whereas for the other groups, their job was not at stake. Rather, they played the role of a potential trainee in participating and evaluating GT-VITA. In an ideal experimental world, actual potential FOT analysts like the computer operators would be the subjects to evaluate GT-VITA for whom it was designed and developed.

Participants' Comments and Experimenter's Observations

The overall response to GT-VITA was tremendous. Every participant showed a lot of enthusiasm about being trained on GT-VITA. Many participants actually commented on how much they looked forward to every session and what "fun" it was to work on a system and learn something too.

Although there were generally no differences in performance among subject groups, there were some observed differences in attitude and idiosyncrasies among subject groups. First, Group B, which was made up of computer operators, were most impressed by GT-VITA and got the most out of the training experience. All expressed their eagerness to learn about what goes on in NASA outside of MSOCC -- where they perform their duties to support mission control. Subject B1 exclaimed during almost every lesson, "This is so neat!" or "This is really good!". During declarative lessons, this subject took her time to read aloud all object or concept explanations.

Another subject in Group B (B3) took the GT-VITA training very seriously even though the nature of the evaluation was explained. He asked for the name of the training course so he could update his resume to include the training participation. He also wanted GT-VITA around so he could "look at the pictures" to refresh his knowledge about the NASA network. Towards the end of training, he also asked if all other DOC operators like himself would be given GT-VITA training. Both subject B2 and B3 (DOC operators) said how they finally understood what it meant to have FOT analysts called them about problems during a real-time support.

Earlier results which showed that subjects in Group B ranked GT-VITA higher than any other group were not surprising. This is the group in which subjects stated at more than one occasion that, "The tutor

is smart!" or "The tutor is strict!". Unlike other groups, subjects in Group B were often doubtful about their performance, especially during the practice lessons. However, they were happily surprised to be able to complete a lesson no matter how difficult it seemed at times. Subject B1 was apologetic about her slower pace when she said reassuringly about the practice lessons, "The system [GT-VITA] is fine. If only I can practice more, I know I will get better." In general, the value of practice was recognized by every participant of GT-VITA; many wished they had the opportunity to complete more practice lessons.

Most of the participants in the other groups viewed the training experience, especially the practice phase, as taking on the tutor's challenge. These participants were driven by the "NASA ego", as subject A1 so aptly called it. Some were reluctant to rely on any paper checklist that was available during the practice phase. Some were adamant about answering questions on GT-POCC without the help of the GT-VITA interface. Subject A3 definitely expressed his intention to "do it all" on his own by refusing any verbal comments from the experimenter with "Shhhhhh". Group D (the software team) subjects explored GT-VITA the most in hope to "crash" the system.

Another noticeable difference was the reaction of subjects in Group B versus most other subjects towards the tutor's interventions during the practice phase. Subjects in Group B generally considered them a sign of their own deficiency in knowledge and skills, with comments such as "I can't do this." On the other hand, most other subjects were annoyed at the "interruptions" in their quest to race the tutor. They reacted with such statements, "I was just about to do that!", "Hey, I didn't get enough time" or "Okay, what now?".

Interestingly, the influence of background experience was evident between subject B3, a DOC operator, and C1, a spacecraft specialist. During a practice lesson, the DOC operator was alerted for missing actions to monitor the spacecraft. The DOC operator said that it was easy for him to forget about the spacecraft because of his automatic emphasis on ground equipment at MSOCC. Conversely, the spacecraft specialist forgot to monitor MSOCC components because, as he explained, "I am a spacecraft person." In another case, the FOT experience of subject C2 was contaminating his interactions on a direct manipulation interface. He said that he had gotten so used to the old consoles with keyboard and control buttons that it was less natural for him initially to think of the NASA system with pictures and objects.

Eventually, subject C2 was more than glad to use these pictures and objects to help him remember what to do in GT-POCC.

Although subject reactions were very positive, there were some common problems that most subjects had when interacting with GT-VITA. First, some subjects complaint about the small font size used for reading explanations. Second, subjects with minimal experience with direct manipulation interfaces had trouble understanding the action sequence representation (i.e., panel-object-value sequence). Third, most subjects were confused between an object label that is clickable versus the object itself. For example, the TAC object on the MSOCC page is inspectable and displays a TAC page showing input/output of TAC channels. Whereas, the TAC label above the object is also inspectable, and when clicked upon displays the control panel of TAC. Students often selected the wrong object as specified by an action sequence, for example. Fortunately, by the time a subject progressed to the practice phase, these three problems represented inconvenience that did not directly affect their real-time performance. However, one phenomenon that did affect subjects' real-time performance is the following. As pointed out by several subjects, they were relying so much on the interface objects and pictures for information and operations that they inevitably forgot about other required control actions or information requests not represented as objects. The VERIFY function and STOL procedural checks were victims of this phenomenon, as evident in the assessment results (see Tables 11-3 and 11-8). The problems discussed so far warrant further investigation and eventual modifications to GT-VITA if necessary.

Subjects also asked about GT-VITA's capabilities. First, many subjects wanted to return to earlier lessons to relearn the declarative material, for example. This is a feature that was conceived for GT-VITA but not implemented. Currently, the lesson to transition could be specified in the session file and/or lesson file by the experimenter. Second, some subjects commented that the graphical interfaces could be useful for on-line aiding and between-support data playback and analysis -- precisely the ultimate goal of the GT-VITA project.

Discussion

The evaluation process for a training system consists of two procedures: developing measures of success and determining the training effect (Goldstein, 1986). Measures of success (criteria) are directly related to the instructional needs that has been identified for the problem domain during the assessment phase. The effect of training, whether during training, on the job, or within the organization, is determined by appropriate experimental design methodologies.

Levels of Criteria

Goldstein discussed four levels of criteria to consider as proposed by Kirpatrick (1959): reaction, learning, behavior, and results. Each level is discussed below in the context of GT-VITA's evaluation study.

Reaction. One way to measure the success of the training system is to assess what the trainees thought of it (Goldstein, 1986). For GT-VITA, the subjects were given questionnaires that record their reactions to GT-VITA in terms of what they learned and the functionalities of the system itself. The reaction data were very promising and could be critical in ensuring the receptivity and continuance of GT-VITA (if fully implemented) or similar instructional programs (Goldstein, 1986).

Learning. Evaluation criteria should also include objective and quantifiable measures of the learning that took place during training (Goldstein, 1986). For GT-VITA, subjects were tested on GT-POCC declarative and procedural knowledge components after the declarative and procedural phases respectively. Although the test was conducted verbally, everyone was asked the same set of questions and had the same resource for seeking answers. Currently, the test results were not directly integrated into GT-VITA. However, during the practice phase, objective assessment data were collected and quantified. Results showed that there were no group difference in the distribution of assessment errors made during real-time practice scenarios. Also, results showed that overall, all real-time activities were completed correctly most of the time without errors. Thus, even though there was a limited number of practice lessons, it seemed reasonable to assume all subjects could achieve the ultimate training goal of completing all activities most of the time over many scenarios.

Behavior. Another measure of success relates to trainees' performance on the job after training. This level is important because superior reactions and performance during training do no necessarily transfer to actual behavior on the job (Goldstein, 1986). The issue of transfer validity was beyond the scope of the current evaluation of GT-VITA. As commented by NASA personnel, due to small sample size (and the accelerated training pace), long term benefits and changes to job performance have yet to be determined with more evaluation studies.

Results. This category of criteria concerns the impact of training results on organizational goals in areas such as cost, turnover and morale (Goldstein, 1986). The evaluation of GT-VITA did not define any criterion explicitly to measure organizational impact. The long term cost justification hypothesized by the tutor/aid paradigm is yet to be investigated. Currently, NASA personnel did speculate that GT-VITA training could reduce training time by at least a month which could translate to significant cost reduction. More evaluation studies will be needed to assess this and other potential impact on NASA.

Pre-Experimental Design

The method adopted to examine the effect of GT-VITA training was the one-group posttest only pre-experimental design (Goldstein, 1986). The method was tailored to GT-VITA training process where all subjects were exposed to the instructional treatment in three phases: declarative, procedural and practice. Posttests were administered after the first two phases while objective assessment data were collected during the third. How valid was this design method?

Validity concerns whether the conclusions of the experiment are well grounded (Adelman, 1991). Two common types of validity examined are internal validity and external validity. Internal validity attempts to establish a causal relationship, that the treatment did make a difference in the outcome of a particular situation (Adelman, 1991; Goldstein, 1986). External validity concerns how well the experimental results generalize to other populations or settings, and it is prerequisited by internal validity (Goldstein, 1986). The general consensus is that a pre-experimental design without pretesting nor a control group violates both internal and external validity (Adelman, 1991; Goldstein, 1986). Specifically, there is no control over threats to internal validity, thus causality cannot be interpreted properly. For instance, the cause of change

during training may be due to how subjects were selected, subjects' reaction to an external event, or subjects' gained experience. So without causality, there can be no generalizability.

For GT-VITA, the question is then whether the declarative and procedural training treatments were responsible for the favorable performances on the posttests. Could it be that some subjects had sufficient prior knowledge to answer the posttests without being trained anyway? The posttest questions were framed to reflect the declarative and procedural components that GT-VITA taught in the context of the GT-POCC system. Thus, it can be argued that even with prior knowledge, a subject had to learn about NASA and its components from the perspective of an FOT analyst specifically within the GT-POCC environment. The following observation supports the argument. Several subjects would "catch" themselves answering a question with their prior FOT knowledge -- "Oops, that answer isn't for this system here" or "No, I learned that somewhere else." These subjects would quickly correct themselves, often referring to GT-VITA interface for answers. Besides the posttests, the subjective performance questionnaires were also designed to isolate the subjects' perception on the effect of GT-VITA on the various knowledge components identified specifically for the GT-POCC system.

For the practice phase, the question is whether GT-VITA was responsible for subjects' ability to perform real-time operator activities, or could there be other explanations that account for their performance? In this case, causality is strongly evident for several reasons. First, subjects had to have learned about individual activities before entering this phase in order to initiate these activities on their own. Second, even if subjects had prior knowledge about FOT operations, they had to learn to perform them within the GT-VITA system interface which was new to every subject. Presumably, it was possible that subjects with more computer experience could traverse through the practice lessons without any GT-VITA training. However, the assessment results showed that no one group did better than another in terms of percentage of errors committed, and that the types of assessment errors committed were consistent among all groups.

In short, although there was no experimental control, other aspects of the GT-VITA evaluation process were successful in compensating for possible threats to internal validity. Given that internal validity can be reasonably established, to what extent can the results for GT-VITA be generalized? Goldstein (1986, p.150) points out that external validity "is always a matter of inference and thus can never

be specified with complete confidence". Since GT-VITA is implemented with the tutor/aid approach and evaluated in the context of the actual task environment, it is hoped that the positive results could potentially apply to other satellite ground control systems and even to other supervisory control systems for which the tutor/aid paradigm is conceived.

In summary, within the scope of GT-VITA, the results reported and discussed here have provided the answers to the questions this evaluation study addressed (see Chapter X). First, GT-VITA does seem to teach the knowledge and skills that an FOT analyst should have. Second, a novice operator can be trained in a reasonable amount of time. Third, GT-VITA is sufficiently adaptive and flexible to accommodate individual styles and pace. Fourth, potential users and training personnel reacted positively to GT-VITA.

CHAPTER XII

CONCLUSIONS

This thesis has detailed an opportunistic approach to training technically oriented adults in complex dynamic systems that unifies recent developments in ITS and emerging research in computer-based operator associates. The tutor/aid paradigm proposes the design of an integrated computer-based support system that serves as a tutor during training, and evolves to an assistant that is present during on-line operations after training. The tutor/aid paradigm hypothesizes that such a system is more cost-effective in the long run, and more motivating for operators to learn, use and trust than two separate tutoring and aiding systems.

The tutor/aid paradigm has been modeled in a proof-of-concept ITS for NASA satellite ground control called GT-VITA which focused on the intelligent tutoring aspect of the paradigmby enhancing the validated OFMspert operator's associate architecture with intelligent tutoring capabilities. OFMspert, and hence, GT-VITA, relies on the Operator Function Model (OFM) for interpreting human performance in supervisory control. GT-VITA embraces many of the characteristics identified within the paradigm for an intelligent tutoring and aiding system. The evaluation study conducted for GT-VITA yielded very promising results. Even in an accelerated mode, at the end, participants were able to learn to control the GT-POCC system through the system interfaces provided by GT-VITA. The flexibility and adaptability of GT-VITA were evident in the objective performance data which showed no group difference. Furthermore, reaction from the participants and NASA personnel at Mission Operations Division was very positive.

Implications of Research

Overall, this research represents a first step towards illustrating the validity and utility of the tutor/aid paradigm for designing an intelligent tutoring system that can potentially function as an operator's assistant during on-line operations of a complex supervisory system. This research assumed a successful computer-based operator's assistant and concentrated on the tutoring end of the tutor/aid paradigm. The paradigm identifies characteristics of the domain knowledge and student model that serve as guidelines for developing such an integrated system. More importantly, the paradigm proposes a pedagogical design framework that structures the training process to prepare a novice operator not only to become a competent supervisory controller, but also to understand the tutor and eventually to use the tutor as an aid. As to how well the student adapts to the transition and how the tutor evolves to an aid represent a natural next step in exploring the validity of the tutor/aid paradigm.

The tutor/aid paradigm is derived from extensive research that examined the issues and problems in training and aiding for operators of complex domains. This research adopts the cognitive instrument view of computers in which they are tools that operators interact with to improve or amplify existing human capabilities. Training and aiding represent two ends of the same spectrum with respect to human-computer interactions. Consequently, many issues in training and aiding are necessarily inseparable: the ultimate goal is to enhance the computer-human team in a symbiotic relationship. Thus, the tenets of the tutor/aid paradigm should be applicable for developing standalone intelligent tutoring systems (without the on-line aiding capabilities), and also be consistent with the design of standalone computer associate systems (without the tutoring functions). The latter point is evident in the computer cooperative problem research conducted recently by Jones (1991).

Even though GT-VITA is implemented for the domain of satellite ground control, the tutor/aid paradigm should apply to other complex domains such as aviation and process control. These predominantly automated systems share many features that are prevalent in complex domains such as

dynamism, numerous interacting components and the supervisory role of the human operator. The tutor/aid paradigm is targeted to this class of systems.

The evaluation study of GT-VITA represents a "non-standard" approach to experimental study. Data collection and analysis were successfully carried out in the absence of a control procedure. Thus, this research shows the value and viability of conducting evaluation studies in a real world context to gather interesting and meaningful data. Such data provide useful insights about the system being evaluated that are unlikely to exist in a controlled, artificial experimental environment.

Finally, this research has proven to be more than an academic endeavor; there is potential technology transfer of ideas that were nurtured and tested in a research environment to the actual task domain of application. More importantly, NASA personnel are planning to field the GT-VITA system as it is at the Mission Operations Division, and to integrate GT-VITA into an overall training program for novice Flight Operations Team analysts. In addition, from a research perspective, the GT-POCC and GT-VITA prototype systems combine to provide a testbed for studying a host of relevant issues including learning, pedagogy, knowledge representation, interface design and intent inferencing. From the NASA perspective, GT-VITA serves an important role in the design of next generation mission control systems by demonstrating viable design methodologies in training and aiding.

In conclusion, this has been a very fruitful research project that contributed to the field of intelligent tutoring system and computer-based training in general. However, as alluded to throughout this thesis, there are many areas of research that can be pursued beyond what is reported here to investigate further the full impact of the tutor/aid paradigm.

Future Research

A list of research areas is described below. The list is by no means an exhaustive one. It is hoped that this thesis will stimulate creative and meaningful research that is only limited by our own imagination.

Training the Task Interface

Currently, GT-VITA is implemented to train operator activities on the interactive, graphical interfaces representing the GT-POCC system. In order to fully prepare novice operators for the actual task environment, GT-VITA must be enhanced with lessons that instruct and provide practice for the simulated GT-POCC task interfaces. For instance, after completing the current practice lessons on the GT-VITA's system interfaces, a student may be introduced to the GT-POCC task interfaces through the lessons of type *Learning Operations by Example* and *Learning Operations by Doing*. Next, the student practices controlling the GT-POCC system with lessons of type *Practicing Operations with Feedback* and *Practicing Operations with Checkpoints*. The pedagogical architecture proposed in this thesis can easily accommodate this enhancement to GT-VITA. For modularity, new LessonObject subclasses for task interface lesson types can be created to encapsulate the data and methods dedicated to processing these lessons.

Transitioning from Tutor to Aid

Issues in the transition from tutor to aid were beyond the scope of this research. Currently, GT-VITA training does not include lessons of type *Performing Operations with Tutor as Aid*. Again, GT-VITA can be enhanced with a new LessonObject subclass for this lesson type. Jones' (1991) research on human-computer cooperative problem solving aptly compliments this phase of training. Jones proposed a suite of tools that were tested on the GT-POCC domain in a system called GT-MOCA. The on-line aiding support tools include an interactive blackboard, graphical visualization of system, and dynamic task allocation capabilities. Ideally, GT-VITA and GT-MOCA can be integrated into one computer support system within the tutor/aid paradigm. Then, during the transition lessons, the student is given the opportunity to learn and practice GT-MOCA capabilities. A research project at Georgia Tech is currently exploring related ideas on the transition from GT-VITA to GT-MOCA (Harris, in progress).

Enhancing the Student Model

There are several avenues to be explored towards more effective diagnosis of student actions. First, a limited-bug library representing common classes of operator errors in GT-POCC operations may assist in

interpreting the sources of assessment errors during practice training. In the current implementation, GT-VITA does not have knowledge about possible reasons for missing actions, for instance. Second, there is a need to integrate much of the data collected on the student during a lessons and between lessons but not utilized. From these data a profile can be built to explicitly reflect the student's performance within lessons and overall competence between lessons. Third, many performance tests can be implemented on-line GT-VITA during or after a lesson. The tests can capitalize on the interactivity of the GT-VITA interfaces as a means for inferring student answers. For example, to test if a student understands a set of failures, the student is required to select the appropriate object or panel to locate the failed component. Such on-line diagnostic tests play a major role in transitioning from one lesson type or another.

Transitioning from Lesson to Lesson

The tutor/aid paradigm proposes a set of lesson types that form the framework for a training program. However, the conditions for transitioning from one lesson to another are not specified. Research on the lesson type transition network (see Figure 4-2) may begin by exploring current conditions of GT-VITA.

Currently, GT-VITA has very limited knowledge about lesson transitions. The repeat option and the next lesson specification in a lesson file determine whether GT-VITA stays on the same lesson or move to the next one. Although the lesson file structure with provisions for specifying the level of difficulty and the next lesson is a flexible means for planning the training curriculum, the process relies on a human instructor or supervisor of GT-VITA. A viable research project is to automate the transition process by taking advantage of diagnostic information from the student model and enhancing GT-VITA with libraries of scenarios and lessons to select from dynamically. For example, each scenario may be annotated with operator knowledge components that the support configurations and scheduled events are designed to teach or test. Other knowledge of lesson transition may be represented in each lesson object such as lesson prerequisites and conditions for repeating the lesson type. For more flexibility and adaptability, students should have the opportunity to decide on lessons to complete, according to their own learning style and

pace. Currently, GT-VITA has an object called StudentModel that basically records a history of all lessons completed. A student may be able to view this history and decide on the next lesson to review, for instance. To what extend lesson transitioning should be student-initiated must be explored in conjunction with the automated process.

Instructor's Interface to the Tutor

No matter how well an instructional system is designed, a human instructor is usually present to initiate the system and to manage any unexpected problems during training. In GT-VITA, a human instructor supervises the training process and initiates the appropriate session for every trainee. The human instructor modifies appropriate set of files to suit the student's state of learning in the training process. One way to facilitate the session initialization is to enhance GT-VITA with a startup interface on which the instructor can access a student profile about how the student has done so far, and dynamically modify the corresponding files. With an automated transitioning process, the student profile may include the tutor's recommended lesson and scenario to use, so that the human instructor may choose to agree or overwrite these recommendations.

Another related idea is authoring tools. That is, besides a means for initialization, an instructor's interface to the computer tutor may be designed to enable tailoring of GT-VITA to specific instructional needs. For example, GT-VITA may be used to trained several missions with different network configurations. The object oriented approach of the GT-POCC system facilitates such domain-specific modifications.

More Extensive Evaluation Studies

The evaluation study reported in this thesis represents the first attempt to do so for GT-VITA. As discussed in Chapter XI, the long term benefits of GT-VITA, and ultimately the tutor/aid paradigm, need to be investigated with further longitudinal studies. Of utmost importance is whether the GT-VITA training has any transfer validity as compared to existing training methods. Also of practical importance is whether the tutor/aid paradigm is cost justifiable in the long run as advocated.

Beyond the Tutor/Aid Paradigm

The development and subsequent success of GT-VITA is due in part to the operator function model and the OFMspert architecture that provide the foundation for GT-VITA. However, the tutor/aid paradigm does not address the selection of the modeling and software architectures necessary to make the tutor/aid system successful. Thus, the jury is still out on whether the tenets of the tutor/aid paradigm could be successfully applied in the absence of OFM and OFMspert. It would interesting to combine other operator performance models and operator's associate architectures to further investigate the utility of the paradigm on the same or different domains of application.

The tutor/aid paradigm also does not address the issue of integrating the tutor/aid support system in the actual task environment. Specifically, in the tutoring mode, the support system has perfect information from the simulation. However, in the on-line aiding mode, the support system, now operational in the actual context, is subjected to the uncertainty and information reliability problems that plague all complex dynamic domains. The tutor/aid paradigm does not address this dilemma. Further research is needed to identify what informational aspects in the tutoring mode can be preserved in the aiding mode, and if not, what the remedies are.

The lesson types proposed within the tutor/aid paradigm are based on the pedagogical matrix of Figure 3-1. A viable research is to examine if other instanciations of the matrix (level of help versus dimension of knowledge), that is, other lesson types, may enhance the pedagogical structure of the tutor. By the same token, whether there may be other instructional strategies for each lesson type or other approaches to realizing the pedagogical matrix should also be examined.

APPENDIX A

GT-VITA'S PEDAGOGICAL STRUCTURE

GT-VITA's pedagogical structure is implemented in an object-oriented approach with two basic object classes: LessonObject and PedagogyModule. There are seven subclasses of LessonObject that capture the seven lesson type proposed in this thesis. Each subclass has a dedicated set of data and methods, and it also inherits all data and methods from its parent class LessonObject. The selection, instianciation and control of a particular LessonObject is supervised by the PedogogyModule object. All student actions are first captured by the PedagogyModule and passed to the appropriate LessonObject from subsequent processing. The PedagogyModule has access to other modules of the GT-VITA's OFMspert system for domain knowledge and diagnostic information. All modules in OFMspert are implemented as subclasses of the super abstract class OFMspertObject.

The class definitions of the PedagogyModule, the LessonObject and its seven subclasses are annotated and presented here. These class definitions include significant data and methods organized by their functionalities for clarity. Most utility and access methods have been ommitted. Relevant supporting structures that a class utilitizes are included at the end of the class definition.

The class definitions are presented in the following order:

PedagogyModule
LessonObject
ProactiveDeclarativeLesson
ModelingDeclarativeLesson
EmpoweringDeclarativeLesson
ModelingProceduralLesson
ProactiveProceduralLesson
ReactiveProceduralLesson
CoachProceduralLesson

GT-VITA's pedagogy relies largely on the methods in the enhanced control environment. This module consists of structures to represent interface panel and object characteristics, and methods to manipulate interface elements for instructional purposes. Currently, these data and methods are not encapsulated in an OFMspertObject. They are presented here following the pedagogical structural definitions. The structural description of the other modules in GT-VITA's OFMspert can be found in Jones, 1991.

```
class PedagogyModule : public OFMspertObject {

/* lesson specific data */
  char* tutorMode;
  char* knowledgeMode;
  LessonObject* currentLesson;
  char lessonFile[30];

/* assess to other tutoring modules */
  StateSpace* stateSpace;
  Blackboard* blackboard;
  ControlledSystemInterface* csi;
  StudentModel* student;

/* data for checkpoint assessments */
  int prepassOkay;
  int onpassOkay;
  int postpassOkay;
  int pendingSetup;
  int lateAPTERM;
  int lateAPSET;
  int reAssess;
  PseudoActionNode* pendingActions;

/* lesson transition methods */
  void initialize();
  void init(int);
  void end();
  void repeat();
  void next();
  int update(char*, char*, int);
  void updateStudentModel();
  int alert(void*, int);
  void eventMsg(char*);
  void emptyPage();

/* feedback/checkpoint assessment methods */
  void evaluateCommand(char*, char*);
  int checkAction(char*);
  int checkRepeatedActions(int, FeedbackStruct*);

  void addPendingActions(FeedbackStruct*);
  void deletePendingAction(PseudoActionNode*, PseudoActionNode*);
  void showPendingActions();
  void evaluatePendingAction(ActionNode*);

  int checkPrepass();
  int checkOnpass();
  int checkPostpass();

  void proceed();
  int tasksCompleted();
  int makeFunctionEC(char*, int, EventContext*);
  void reAssessBlackboard(char*);
  int reAssessFeedback(FeedbackStruct*);
};
```

```
class LessonObject : public OFMspertObject {

  LessonObject* next; // a linklist of lesson objects
  PedagogyModule* tutor; //the tutor object supervising lesson objects

/* lesson identification data */
  char id[3];
  char* tutorMode;
  char* knowledgeMode;
  char** lessonId;
  char** lessonView;
  char** lessonTarget;

/* explanations data */
  char** goal;
  char** instructions;
  char** lessonExplanation;
  char** supportExplanation;
  char** failuresExplanation;
  char** startInstructions;

/* system failures data */
  Failure* failuresTable;
  Failure* currentFailure;
  int failIndex;
  int totalFailures;

/* lesson transition data */
  PedagogyModule* tutor;
  char* nextLessonFile;
  char* nextLessonType;
  int difficulty;
  int status; //of diagnosis for student model;
  int resumeNext;
  int resumeRepeat;

/* scenario data */
  Scenario* currentScenario;
  int startScenario;
  int nextScenario;
  int scenarioStartTime;
  int scenarioStopTime;

/* viewing frequency data */
  int viewGoal;
  int viewExplanation;
  int viewInstruction;
  int viewSupport;
  int viewExplainSupport;
  int viewFailure;
  int viewExplainFailure;
  int viewStart;

/* lesson setup/ending methods */
  void initialize();
  void setup();
  int lessonReady();
  void closePanels();
```

```
  int fastforward(int);
  void initSubjdata(char*);
  void endSubjdata();

/* support scenario methods */
  void initSupportExplanation();
  void updateSupportInfo();
  void updateSupportPanel(char*);
  void updateExplainSuppPanel(char*);
  int scenarioStarted();
  int updateScenario();

/* system failures methods */
  void initFailures();
  void initFailureExplanation(char*, Failure*);
  void updateFailuresInfo();
  void updateFailuresPanel(char*);
  void updateExplainFailPanel(char*);
  Failure* getFailureUnit(char*);

/* communication methods for the lesson panel */
  void update(char*, char*, int);
  int processMiscAction(char*);
  void processLessonPanel(char*, char*);
  void updateGoalPanel(char*);
  void updateExplanationPanel(char*);
  void updateInstructionsPanel(char*);
  void updateStartPanel(char*);

/* communication and instructional methods defined at
    the subclass level */
  virtual void init(FILE*);
  virtual void evaluateSelection(char*, char*);
  virtual void evaluateTargetAction(char*, char*);
  virtual void evaluateDialogAction(char*, char*);
  virtual void evaluateControlAction(char*, char*);

  virtual void modelFailure(char*);
  virtual void alert(void*, int);
  virtual void startReplay(char*);
  virtual void report();
};



/* a system failure structure */
class Failure {
  public:
   char* event;
   char** explanation;
   char** panels;  // panels associated with failure
   int* panelViewed; //have panels been viewed?
   int viewed; // has failure been studied
   int done; // is failure done
   Failure* next;
};
```

```
class ProactiveDeclarativeLesson : public LessonObject {

/* lesson specific data */
  PDlesson* lessonUnitsTable;
  PDlesson* currentUnit;
  char* currentSelection;
  int displayIndex;

/* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void reset();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateProactivePanel(char*);
  void updateObjectsPanel(char*);
  void updateExplainPanel(char*);
  int processItem(char*);

};



/* a lesson unit structure */
class PDlesson {
  public:
   char* panel;
   int flag; // flag=1 if panel is to stay when moving on to next panel
   char* parent;
   char* parmNames; // list of parmNames
   int* selCount; // counting each parmName selection
   char** explanation;
   int viewed; //panel explanation viewed?
   PDlesson* next;
   PDlesson* previous;
};
```

```
class ModelingDeclarativeLesson : public LessonObject {

/* flow explanations */
  char** systemDataflow;
  char** gsfcDataflow;
  char** msoccDataflow;
  char** tacDataflow;
  char** spacecraftDataflow;
  char** cdhsDataflow;
  char** rfsDataflow;

/* frequency of viewing flow explanations */
  int viewSystemFlow;
  int viewGsfcFlow;
  int viewMsoccFlow;
  int viewTacFlow;
  int viewSpacecraftFlow;
  int viewCdhsFlow;
  int viewRfsFlow;

/* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateModelPanel(char*);
  void processFlow(char*, char*, char*);
  void modelFailure(char*);

};
```

```
class EmpoweringDeclarativeLesson : public LessonObject {

/* lesson specific data */
  char** lessonExplanation;
  ObjectsDispatch* currentObject;
  int displayIndex;
  int inspectCode;
  int hierarchyShown;
  CodeViewed* codeCounts;

 /* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateCodePanel(char*);
  void updateHierarchyPanel(char*);
  void updateInspectPanel(char*);
  void updateCodeCount(char*, char*);
  void addCodeCount(char*, char*);
  CodeViewed* findCodeCount(char*, char*);

};


/* an interface object structure */
struct ObjectsDispatch {
   char* panel;
   char* parmName;
   char* label;
   char* type;
   float defaultValue;   // for type "real"
   float highlightValue; // for type "real"
   char* nextPanel;
   char** childobjects;
   char* file; // include file of class definition
   struct ObjectsDispatch* next;
};

/* a structure to keep track of inspected objects */
struct CodeViewed {
  char* panel;
  char* parmName;
  int codeViewed;
  struct CodeViewed* next;
};
```

```
class ModelingProceduralLesson : public LessonObject {

/* lesson specific data */
  char** commandlist; // for textlist
  char** actionlist; // for textlist
  char** commandExplanation;
  char** actionsExplanation;
  char* currentCommand;
  Command* currentCommandStruct;
  Action* currentAction;

  int currentTime;
  int commandIndex;
  int actionIndex;
  int viewActions;
  int pendingReset;

/* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void reset();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateReplayPanel(char*);
  void updateCommandPanel(char*);
  void updateActionsPanel(char*);
  void updateExplainCommandPanel(char*);
  void updateExplainActionsPanel(char*);
  void processItem(char*);

  void startReplay(char*);
  void modelFailure(char*);
  int replayCommand();
  int replayAction();
  int showActionSequence(int);
  void commandCompleted();
  void showExplainAction(char*);

};

/* structures defined in the next lesson type */
class Command;
class Action;
```

```
class ProactiveProceduralLesson : public LessonObject {

/* lesson specific data */
  TaskStruct* tasks;
  TaskStruct* currentTask;
  TaskStruct* pendingTasks;
  Unit* currentStep;
  Unit* currentProblem;
  Command* currentCommandStruct;
  int currentStepIndex;
  int ready;
  int teachProblems;

/* lesson transition methods */
  void init(FILE*);
  void readSteps(TaskStruct*);
  void readProblems(TaskStruct*);
  void readStepExplanation(Unit*);
  void readProblemExplanation(Unit*);
  void setup();
  int diagnose();
  void reset();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateSelection(char*, char*);
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateProcTaskPanel(char*);
  void updateStepsPanel(char*);
  void updateProblemsPanel(char*);
  void processStepHelp(char*);
  void processProblemHelp(char*);
  void updateExplainStepPanel(char*);
  void updateExplainProblemPanel(char*);
  void undoItemSelection(char*, char*);

  void alert(void*, int);
  void instructTask(TaskNode*);
  void modelFailure(char*);
  void updateCommandStatus();
  void updateTaskHistory();
  int tasksCompleted();
  void determineName(Unit*, char*);
  int verifyAction(char*, char*, int);
  void addToPendingTasks(TaskNode*);
};
```

```
/* a structure for a unit step or problem for a task */
class Unit {
  public:
   char* name; // AP3 init
   char* abbrev; //APSET
   char** explanation;
   int done;
   Unit* next;
   Unit* previous;
};

/* a structure for a task to be instructed */
class TaskStruct {
  public:
   char* name; // name of task
   char** explanation;
   int viewed; //task explanation viewed?
   Unit* steps;
   Unit* problems;
   int numSteps;
   int numProblems;
   TaskNode* bbnode;
   TaskStruct* next;
   TaskStruct* previous;
   int time;
   int done;
};


/* a structure for representing the action sequences of
    a command */
class Command {
 public:
   char** commands;  //array of possible commands with same actions
   Action** actionSequences;  // array of possible action sequences
   Command* next;
};

/* a structure for representing an action */
class Action {
  public:
   char* panel;
   char* parmName;
   char* value;
   Action* next;
   Action* previous;
}
```

```
class ReactiveProceduralLesson : public LessonObject {

/* lesson specific data */
  AlertItem* pendingItems;
  AlertItem* currentAlert;
  int alertPending;
  int alertActionCount;
  int alertFeedbackCount;
  int prepassError;
  int onpassError;
  int postpassError;

/* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void report();
  void closePanels();

/* communication and instructional methods */
  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateReactivePanel(char*);
  void updateAlertPanel(char*);
  void updateExplainAlertPanel(char*);
  void processUnconnectedAction(ActionNode*);
  void processFeedback(FeedbackStruct*);
  void explainFeedbackAction();

  void alert(void*, int);
  void evaluateAlertActions(char*, char*);
  int okayToAlert(FeedbackStruct*);

  void addToPendingList(void*, int);
  void addPrepassError();
  void addOnpassError();
  void addPostpassError();
};



/* an alert item structure */
class AlertItem {
 public:
  void* structure;
  int type;
  int attended;
  int okayToAlert;
  AlertItem* next;
};
```

```
class CoachProceduralLesson : public LessonObject {

/* lesson specific data */
  int prepassError;
  int onpassError;
  int postpassError;

/* lesson transition methods */
  void init(FILE*);
  void setup();
  int diagnose();
  void report();
  void closePanels();

/* communication and instructional methods */

  void evaluateTargetAction(char*, char*);
  void evaluateControlAction(char*, char*);
  void evaluateDialogAction(char*, char*);

  void updateCoachPanel(char*);
  void explainFeedbackAction();
  void alert(void*, int);

  void addPrepassError();
  void addOnpassError();
  void addPostpassError();

};
```

```
/************ The Enhanced Control Environement ************/

/*  structure  to  represent  panel  characteristics  */
struct PanelsDispatch {
    char* name;
    char* label;
    Id* panelId;
    Id* panelView;
    Id* panelTarget;
    struct PanelsDispatch* next;
};

/*  structure  to  represent  object  characteristics  */
struct ObjectsDispatch {
   char* panel;
   char* parmName;
   char* label;
   char* type;
   float defaultValue;   // for type "real"
   float highlightValue; // for type "real"
   char* nextPanel;
   char** childobjects; /* for "composite" objects */
   char* file; // file of class definition
   struct ObjectsDispatch* next;
};

/*  the  panels  and  objects  tables  defined  */
PanelsDispatch* panelsTable;
ObjectsDispatch* objectsTable;

/*  methods  to  create  the  tables  */
addPanelsTable(char* name, Id* panelId, Id* panelView, Id* panelTarget)
makeObjectsTable()
setChildobjects(ObjectsDispatch* aStruct)

/*  methods  to  assess  an  object  or  a  panel  by  its  name  */
Id* getPanelTarget(char* name)
Id* getPanelView(char* name)
Id* getPanelId(char* name)
ObjectsDispatch* getObjectStruct(char* panel, char* parmName)
char* getLabel(char* panel, char* parmName)
char* getPanelLabel(char* name)

/*  methods  to  manipulate  panels'  and  objects'  interface
    representation  */
processTargetAction(char* panel, char* parmName, int help)
highlightObject(char* panel, char* parmName)
unhighlightObject(char* panel, char* parmName)
isHighlighted(char* panel, char* parmName)
isShown(char* panelName)
highlight(panelId)
unhighlight(panelId)
reverseItemColor(Id panelId, Id panelView, char* itemName)
int isMapped(panelId)
int displaySide(panelId)
int isWidgetColorChanged(panelId, panelView, itemName)
changeWidgetColor(panelId, panelView, itemName, fgname, bgname)
reverseWidgetColor(panelId, itemName)
```

APPENDIX B

GT-VITA INSTRUCTIONS

Introduction to the GT-POCC System and the GT-VITA System

The Georgia Tech Payload Operations Control Center (GT-POCC) is a simulated environment of the NASA data/information system for the capture and processing of data from near-earth scientific satellites. In GT-POCC, you are a member of the Flight Operations Team for a hypothetical spacecraft called GASP (Goddard Atmospheric Space Platform). The Geogia Tech Visual and Inspectable Tutoring and Aiding (GT-VITA) System is a computer-based training system that uses the GT-POCC environment to teach you about components in the NASA system and also various duties that you are responsible for as a Flight Operations Team (FOT) analyst.

GT-VITA displays various entities in the NASA data/information system as graphical objects and pictures on two computer screens. To help you learn about the NASA system and your role in controlling it, GT-VITA lets you interact directly with the objects and pictures in the on-screen NASA environment. A special feature of the GT-VITA display is that it can be animated, allowing you to observe the action as data flows through the NASA network. The goal of the GT-VITA tutor is to guide you through interactions with the on-screen NASA enviroment, lesson by lesson, gradually building up your knowledge of the NASA system and your proficiency in controlling it.

To help you navigate the GT-VITA tutoring system, instructions are provided for each lesson type. Each set of instructions describes what you will see over the course of the lesson, the actions you can take, and what the tutor will do in response. Sample snapshots of the on-screen environment are included with each instruction set to help you find your way through the lesson. With help from on-screen and off-screen sources, you should be well-equipped to successfully complete GT-VITA training.

(Note: Figures have been ommitted in the subsequent reproduction of the lesson instructions; they are redundant with those in the main text).

Lesson Type 1: Learning System Components

The main panel of this first lesson contains several labeled buttons that can be "clicked on" and activated with the mouse cursor. To click on an object on the screen, move the mouse to position the screen cursor on top of the object, and then "click" the left button of the mouse.

First the top row of buttons [see snapshot 1]:

Goal - Clicking the mouse on the **Goal** button causes a panel to appear that explains the goal of this lesson. Note that the **Goal** button changes color when it's corresponding panel is open. This new Lesson Goal panel is typical of the panels of the tutor. It has a title bar at the top, and within the panel is important information or notices. If the text inside is too long to fit within the panel, a scroll bar along the side of the panel allows you to move the text (click on the arrows or the dark area between them). The tutor can't see when you have finished reading the information you need from a panel, so to acknowledge that you have finished, click on the small box in the upper-left corner of that panel before continuing. Clicking on this box closes the panel. You can always refer back to the Lesson Goal later by clicking on the **Goal** button.

Instructions - Clicking the mouse on the **Instructions** button causes a panel to appear that explains how you can interact with the tutor to discover the information presented in this lesson. Confirm that you've read the Lesson Instructions by clicking on the small box in the upper-left corner of the Instructions panel to close the panel before continuing. These instructions can always be referred later on if you need them simply by clicking on the **Instructions** button once more.

The top row of buttons in the main panel is consistent across all lessons. The first thing to do when beginning any lesson is to click on each button of the top row and read the text in the panels that appear. Remember to close these panels when you finish reading before continuing. If you try to continue with the lesson without first clicking on each button of the top row, a miniature alert panel will remind you - just click the **Noted** box of the alert panel to continue [see snapshot 1]. Throughout the tutoring session, whenever an alert panel appears, no other actions can take place until the alert is acknowledged by clicking its **Noted** box.

The second row has a single button:

Target Panel Explanation - This button opens a panel that explains what will be displayed and discussed in this section of the lesson. As you finish each section, you'll need to use to this button again to open a new Target Panel Explanation.

The most used button of this lesson:

Objects List - This button opens a panel listing objects to be studied in this section [see snapshot 2]. To get a description of each object in the list, either click in the checkbox beside its name, or click on the picture of the object in the graphics panel. When you have finished looking at the descriptions of all the objects in the list, click the **Continue** button in the Objects List panel to move on to the next section of the lesson, which will be about another group of objects.

To start over and study the current group of objects again, click on the **Quit** button in the Objects List panel.

Before going through each section's **Object List**, remember to check the **Target Panel Explanation**. If you forget to do so, a miniature alert panel will remind you - just click the **Noted** box of the alert panel to continue.

Lesson requirements:

When you have looked at each group of objects, an alert panel will appear letting you know that you can either **Repeat** the current lesson or **Continue** to the next lesson. Confirm that you have read this message by clicking on the **Noted** box, and the alert will disappear and let you continue.

Repeat - If you want to look at the lesson again click the **Repeat** button. All lessons of the tutor have a **Repeat** button that can be used as often as needed.

Continue - If you want to go on to the next lesson, click the **Continue** button.

Quit - If, for some reason you wish to exit the tutor, click on the **Quit** button which is positioned in the top-right corner of each lesson's main panel.

Lesson Type 2: Learning System Behavior

As in previous lessons, the first thing to do is to click on each button of the top row of the main panel: **Goal, Explanation,** and **Instructions** [see snapshot 1]. Remember to close the informational panels after reading the text inside.

This type of lesson describes and animates what happens during a routine spacecraft support, so the next two buttons, **Support Configuration** and **System Failures**, open up to specifics of the support scenario.

Support Configuration - The panel this button opens contains specific information about the spacecraft support; the kind of information that would be on a Pass Plan for this support [see snapshot 2]. If you click on the **Explain** button at the top of the Support Configuration panel, a new panel will open with details about the support scenario. When you have finished reading the Support Configuration panels, remember to confirm by closing them. You can refer to this information anytime during the lesson by clicking on the **Support Configuration** button later.

System Failures - The panel this button opens lists the failures that will happen during the support scenario [see snapshot 3]. Clicking on the **Explain** button of the System Failures panel opens up a general description of the types of failures you are about to see. When you have finished reading the System Failures panels, remember to confirm by closing them. You will want to open the System Failures panel again to learn about specific failures as they happen during the scenario.

Start Scenario - The panel this button opens is your window to view the inner workings of a spacecraft support [see snapshot 4]. Read the instructions for the scenario, and click the **Show NASA Network** button in this panel to bring up a picture of the NASA network. If you click the **Start** button, the scenario will begin and you can see data flowing through the system between the spacecraft and the ground facilities. If you want to freeze the activity, the **Pause** button will stop the clock until you are ready to **Start** again.

Because a goal of this type of lesson is to demonstrate how data flows through the NASA network and how different failures affect that flow, the two critical buttons to use are **System Failures** and **Explain Flow**.

Explain Flow - This button opens a menu list of different NASA information flow paths. To select a path from the menu, click on **Explain Flow** and hold the mouse button down as you drag down - release the mouse button when the cursor is positioned over the desired flow path (this mouse operation works for any pulldown menu; they are marked with an arrowhead symbol under the label). When you select a flow path to view, an explanation panel will open beside the pictures of the objects being discussed [see snapshot 5]. Another way to view data flows is to click directly on the objects involved. Remember to close the explanation panels when you finish reading them.

As failures happen during the support scenario, refer to **System Failures**. Each failure event will be highlighted in the panel's list as it occurs. Read details of the highlighted event and how to view it by clicking on the **Explain** button. You can highlight failure events yourself by clicking the mouse on an event in the list. Click directly on pictures of objects to see how failures affect the parts of the NASA network associated with each failure event.

Lesson requirements:

Before the lesson ends, you should view each data flow explanation and explanations of each System Failure event. Alert panels will tell you if you missed a piece of information along the way. When the scenario is finished and you have viewed each explanation, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson.

Repeat - If you would like to see new support scenarios and how they affect the NASA network, click the **Repeat** button to do this type of lesson again.

Continue - Click the **Continue** button if you would like to try a new type of lesson.

Lesson Type 3: Exploring Tutor's Knowledge

This type of lesson is designed to let you explore the components of the NASA network and how they function and interact during a typical spacecraft support scenario.

As always, the first thing to do is to open up each button on the top row of the main panel: **Goal**, **Explanation**, and **Instructions** [see snapshot 1]. Remember to close these informational panels after reading the text inside.

Support Configuration - The panel this button opens contains specific information about the spacecraft support; the kind of information that would be on a Pass Plan for this support. If you click on the **Explain** button at the top of the Support Configuration panel, a new panel will open with details about the support scenario. When you have finished reading the Support Configuration panels, remember to confirm by closing them.

Start Scenario - Once more, the panel this button opens is your window to view the mechanics of a spacecraft support. Read the instructions for the scenario, and click the **Show NASA Network** button in this panel to bring up a picture of the NASA network. If you click the **Start** button, the scenario action will begin. If you want to freeze the activity, the **Pause** button will stop the clock until you are ready to **Start** again.

Explore the NASA network by clicking on pictures of objects in the network - some open into more detailed views and perspectives. If you wish to see a written explanation the objects you click on, click on the **Inspect Representations** checkbox in the main panel.

Inspect Representations - When this checkbox is activated, clicking on pictures of objects causes panels to open that explain how the objects you click on fit into the hierarchy of objects in the NASA network [see snapshot 2]. Each of these panels has a **Class Hierarchy** button in its title bar. Clicking on **Class Hierarchy** causes another panel to appear that shows the entire hierarchy of objects represented in the support simulation. You can deactivate the Inspect Representation feature by clicking a second time on the **Inspect Representations** checkbox.

When the current scenario is finished, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson. This lesson has no specific requirements, so you can fast-forward to the next scenario with the **Repeat** button, or **Continue** at any time during the scenario.

Repeat - If you would like to see new support scenarios and how they affect the NASA network, click the **Repeat** button to do this type of lesson again.

Continue - Click the **Continue** button if you would like to try a new type of lesson.

Lesson Type 4: Learning Operations by Example

As in previous lessons, the first thing to do is to click on each button on the top row of the main panel: **Goal, Explanation**, and **Instructions** [see snapshot 1].

The second row of buttons is also consistent with previous lessons:

Support Configuration - The panel this button opens contains specific information about the spacecraft support you will see; the kind of information that would be on a Pass Plan for this support. When you have finished reading the Support Configuration panel and its accompanying Explanation panel, remember to close them.

System Failures - The panel this button opens lists the failures that will happen during the support scenario. When you have finished reading the System Failures panel and its accompanying Explanation panel, remember to close them. You will want to open the System Failures panel again to learn about specific failures as they happen during the scenario.

To watch a support scenario in action and see how the NASA network is commanded from the Mission Operations room, use the **Start Scenario, Operator Commands**, and **System Failures** buttons.

Start Scenario - Read the instructions for the scenario, and click the **Show NASA Network** button in this panel to bring up a picture of the NASA network. If you click the **Start** button, the scenario will begin. If you want to freeze the activity, the **Pause** button will stop the clock until you are ready to **Start** again.

Operator Commands - The panel this button opens contains a list of operator commands and the times they are due to take place [see snapshot 2]. The moment a command from the Mission Operations Room is supposed to happen, the scenario clock will stop, allowing you to watch the action step-by-step. If you want to see the command executed all at once, click the **Execute Command** button. If you want to see the sequence of steps involved in the command activity, click the **Show Action Sequence** button. Because the scenario clocked stopped so that you could view the current command, use the **Continue** button to restart the clock and go on to the next command.

Show Action Sequence - The panel this button opens lists the objects involved in the execution of the current command, and which graphic panels their pictures can be found in [see snapshot 2]. Use the **Step** button to see the action happen step-by-step, or use the **Execute All** button to see the action happen all at once. Because the scenario clocked stopped so that you could view the current command, use the **Continue** button to restart the clock and go on to the next command. If there is more than one sequence of panels for seeing the command activated, clicking the **Others** button will show an alternate list of graphic panels and objects.

As failures happen during the support scenario, refer to **System Failures**. Each failure event will be highlighted in the panel's list as it occurs. Read details of the highlighted event and how to view it by clicking on the **Explain** button. You can highlight failure events yourself by clicking the mouse on an event in the list. Click directly on pictures of objects to see how failures affect the parts of the NASA network associated with each failure event.

Lesson requirements:

Before the lesson ends, you should view each command and explanations of each System Failure event. Alert panels will let you know if you missed a piece of information along the way. When the scenario is finished, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson.

Lesson Type 5: Learning Operations by Doing

As always, the first thing to do is to click on both buttons on the top row of the main panel: **Goal** and **Instructions.**

This lesson is divided into several sections, each guiding you through completion a different Mission Operations Room (MOR) task. The first step when beginning each task section is to click on the **Task Explanation** button:

Task Explanation - This button opens a panel that explains which MOR task will be discussed in this section of the lesson.

The next row of buttons is familiar from previous lessons:

Support Configuration - The panel this button opens is like a Pass Plan of the support scenario. When you have finished reading the Support Configuration panel and its accompanying Explanation panel, remember to close them.

System Failures - The panel this button opens lists the failures that will happen during the support scenario. When you have finished reading the System Failures panel and its accompanying Explanation panel, remember to close them. You will want to open the System Failures panel again to learn about specific failures as they happen during the scenario.

To begin a support scenario:

Start Scenario - Read the instructions for the scenario, and click the **Show NASA Network** button in this panel to bring up a picture of the NASA network. As before, the **Start** button begins the scenario, while the **Pause** button will stop the scenario clock until you are ready to **Start** again.

Alert panels will notify you during the course of the scenario when it is time to perform some Mission Operations Room task [snapshot 1]. The scenario clock will stop while you check the **Task Explanation** and click on **Steps in Task**:

Steps in Task - This button opens up a panel listing steps for the task to be studied in this section of the lesson [snapshot 2]. To get help accomplishing each action, click in the Help checkbox beside the action's name. A panel will open that lists, in sequence, the NASA network objects you should click on to perform the action [snapshot 3].

When you have finished doing all the task steps, click on the **Continue** button in the Steps in Task panel to restart the scenario clock and the NASA network action.

To clear the task checklist and study the current task again, click on the **Quit** button in the Steps in Task panel. Make sure you understand each task before continuing.

In addition to commanding the NASA network directly by clicking on the pictures of NASA network components, the system can be controlled by **Other Actions**:

Other Actions - This button opens a pulldown menu of control actions [snapshot 4]:

Bookkeeping - This option opens a panel of Bookkeeping Actions the FOT analyst is responsible for.

Communications - This option opens a panel of Communications with other facilities.

Pre-Pass Verification - This option opens a panel of Verification Actions the FOT analyst is responsible for during the "pre-pass phase" before the actual support begins.

STOL Procedures - This option opens a panel of STOL Procedures the FOT analyst is responsible for during the "on-pass phase" (the period of actual contact with the spacecraft).

Status Information - This button opens a pulldown menu of information sources used in the Mission Operations Room [snapshot 5]:

System Events - The Events Page logs feedback from your control actions and messages from other facilities in the NASA network.

Support Schedule - This option opens a panel listing the spacecraft supports you will be commanding.

History - This button opens a pulldown menu that lists all the tasks you have completed up to the current point in time.

Lesson requirements:

Before the lesson ends, you should carry out each task and read explanations of each failure event. Alert panels will let you know if you missed a piece of information along the way. When the scenario is finished, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson.

Lesson Type 6: Practicing Operations with Immediate Feedback and Checkpoints

As always, the first thing to do is to click on buttons on the top row of the main panel: **Goal**, **Explanation**, and **Instructions**. Remember to acknowledge you have read the information by closing each panel.

This lesson type is one in which you are responsible for monitoring and commanding the NASA network during the scenario. The tutor will act as a "safety net", catching you if you miss a cue and letting you know how get back on track.

Support Configuration - This is your script for the current support [snapshot 1]. Read it carefully for information about what to expect during each support.

In addition to commanding the NASA network directly by clicking on the pictures of NASA network components, the system can be commanded through **Other Actions**:

Other Actions - This button opens a pulldown menu of special control actions:

Bookkeeping - This option opens a panel of Bookkeeping Actions for which the FOT analyst is responsible. If the Bookkeeping Action you select ends in ellipsis ("..."), click the **Details** button beside it to follow through with that action completely. The action you choose is activated by clicking the **Okay** button (or cancelled by clicking the **Cancel** button).

Communications - This option opens a panel of Communications with other facilities that the FOT analyst is responsible for. If the Communications action you select ends in ellipsis ("..."), click the **Details** button beside it to follow through with that action completely. The action you choose can be activated by clicking the **Okay** button, or cancelled by clicking the **Cancel** button.

Pre-Pass Verification - This option opens a panel of Verification Actions the FOT analyst carries out during the "pre-pass phase" before the actual support begins. The action you choose is activated by clicking the **Okay** button (or cancelled by clicking the **Cancel** button).

STOL Procedures - This option opens a panel of STOL Procedures the FOT analyst carries during the "on-pass phase" (the period of actual contact with the spacecraft).

The procedure you choose is activated by clicking the **Okay** button (or cancelled by clicking the **Cancel** button).

Status Information - This button opens a pulldown menu of information sources [snapshot 2]:

System Events - This option opens the System Events Log. Feedback from your control actions and messages from other facilities in the NASA network are logged on this page.

Support Schedule - This option opens a panel listing the spacecraft supports you will be commanding.

Performance Assessments - Selecting this option is the best way to find out what actions you might have missed (or done at an inappropriate time) during the scenario [snapshot 3]. The important tutor assessments are:

All uninterpreted actions - This panel lists the actions that were unexpected by the tutor. When this happens, you should highlight the unexpected action code and click **Explain** to obtain more details.

Action assessments - This panel lists the tutor's assessments of actions to expect during the scenario [snapshot 4]. If you missed an action or repeated an action needlessly for a particular task (you will see "no" beside the task's assessment code), you should highlight the assessment and click **Explain** to obtain more details. The Assessment Explanation panel lists any missed actions. If you need help performing an action, highlight it and click **Explain**, and step-by-step instructions appear.

Action interpreter (ACTIN) - This panel displays a total perspective with "trees" of tasks the tutor is expecting. Actions you make are shown at the roots of the trees. Click **Explain** for details of how ACTIN works.

After you've gotten your bearings and are ready to monitor and take care of orbiting spacecraft, click **Start Scenario** to begin the action.

Start Scenario - When you're ready, **Show NASA network**, click the **Start** button, and the scenario action will begin. If you want to freeze the activity, the **Pause** button will stop the clock until you are ready to **Start** again.

During the course of the scenario, if you happen to get into trouble, you will hear a beep and the **ALERT** button on the main panel will light up. At this point, the scenario clock will halt, and you should stop what you are doing and click on **ALERT**.

ALERT - The panel this button opens explains what went wrong. Look for further information using the **Performance Assessment** menu option under **Status Information.**

An uninterpreted action means you should check under **Uninterpreted Actions** and click **Explain** for that action [snapshot 5]. An uncompleted or repeated action means you should check under **Action Assessments**, get the Explanation for that action [snapshot 6], and perform any uncompleted action.

After you've thoroughly responded to **ALERT**, you can restart the scenario clock by clicking **Continue** on the main panel. At the point where there are no more **ALERT**s and the button turns gray again, you can continue monitoring and controlling the network.

Pending Actions - This button only appears at critical points in the support (for example, pre-pass to on-pass phase transitions) when there are uncompleted actions pending [snapshot 7]. The scenario clock will stop while you catch up. As you finish each pending action, it will be erased from the Pending Actions list. When you finish them *all, the* ***Pending Actions*** *button will* disappear, and the clock will restart.

Lesson requirements:
When the entire scenario is finished, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson.

Repeat - If you would like to see new support scenarios and how they affect the NASA network, click the **Repeat** button to do this type of lesson again.

Continue - During the scenario, whenever the clock has stopped for an **ALERT** and you have fully investigated the explanation for the ALERT, clicking on **Continue** will restart the scenario clock. At the end of the scenario, click the **Continue** button if you would like to try a new type of lesson.

Lesson Type 7: Practicing Operations with Checkpoints Only

As always, the first thing to do is to click on buttons on the top row of the main panel: **Goal**, **Explanation**, and **Instructions**. Remember to acknowledge you have read the information by closing each panel.

This lesson type is one in which you are responsible for monitoring and commanding the NASA network during the scenario. The tutor will interrupt you only if you don't complete tasks before critical points of the scenario (just before moving from pre-pass phase to on-pass phase, for instance).

Support Configuration - This is your script for the current support [snapshot 1]. Read carefully for information about what to expect during each support.

In addition to commanding the NASA network directly by clicking on the pictures of NASA network components, the system can be commanded with **Other Actions**:

Other Actions - This button opens a pull-down menu of special control actions:

Bookkeeping - This option opens a panel of Bookkeeping actions for which the FOT analyst is responsible. If the action you select ends in ellipsis ("..."), click the **Details** button beside it to follow through with that action completely. The action you choose is activated by clicking the **Okay** button (or cancelled by clicking the **Cancel** button).

Communications - This option opens a panel of Communications with other facilities for which the FOT analyst is responsible. The action you choose can be activated by clicking the **Okay** button, or cancelled by clicking the **Cancel** button.

Pre-Pass Verification - This option opens a panel of Verification Actions the FOT analyst carries out during the pre-pass phase of the support.

STOL Procedures - This option opens a panel of STOL Procedures the FOT analyst carries out during the on-pass phase of the support.

Status Information - This button opens a pulldown menu of information sources:

System Events - This option opens the System Events Log.

Support Schedule - This panel lists the spacecraft supports you will be commanding during the scenario.

Performance Assessments - Opening this panel is the best way to find out what actions you might have missed (or done at an inappropriate time) during the scenario. The important tutor assessments are:

All uninterpreted actions - This panel lists the actions that were unexpected by the tutor. When this happens, you should highlight the unexpected action code and click **Explain** to obtain more details.

Action assessments - This panel lists assessments of expected actions. If you missed or repeated an action during the scenario, you should highlight the assessment and click **Explain** to obtain more details. The Assessment Explanation panel lists any missed actions. If you need help performing an action, highlight it and click **Explain**, and step-by-step instructions appear.

Action interpreter (ACTIN) - This panel displays a total perspective with "trees" of the tasks the tutor is expecting. Actions you make are shown at the roots of the trees. Click **Explain** for details of how ACTIN works.

After you've gotten your bearings and are ready to take care of orbiting spacecraft, click **Start Scenario** to begin the action.

Start Scenario - Whenever you're ready, **Show NASA Network**, click the **Start** button and the scenario action will begin. If you want to freeze the activity, the **Pause** button will stop the clock until you are ready to **Start** again.

Pending Actions - This button only appears at critical points in the support (for example, pre-pass to on-pass phase transitions) when there are uncompleted actions pending [snapshot 2]. The scenario clock will stop while you catch up. As you complete each pending action, it will disappear from the Pending Actions list. When you finish them all, the **Pending Actions** button will disappear, and the clock will restart.

Lesson requirements:

When the entire scenario is finished, an alert panel will appear letting you know that you can either **Repeat** the current lesson type or **Continue** to the next type of lesson.

APPENDIX C

GT-VITA KNOWLEDGE QUESTIONNAIRES

GT-VITA Knowledge Questionnaire: GT-POCC Objects and Relations

Date: _____

Analyst: _____

1. What are the major components in the NASA Goddard information/data system?

a) _____ confidence level

b) ___ Spacecraft ___ TDRSS ___ GSFC

___ White Sands ___ ground network

2. What are the major components in the Goddard Space Flight Center (GSFC) ground control system?

a) _____ confidence level

b) ___ Nascom ___ MSOCC ___ NCC ___ SPDF

3. What are the major components in MSOCC?

a) _____ confidence level

b) ___ Nascom ___ TAC ___ AP ___ RUPS

4. What are the major TAC components?

a) _____ confidence level

b) ___ A channel ___ B channel

5. What are the major spacecraft components in the GT-POCC?

a) _____ confidence level

b) ___ RFS ___ ACS ___ CDHS ___ PS

6. In the radio frequency spacecraft subsystem, what components do the FOT monitor?

a) _____ confidence level

b) ____ attennas ____ transponders

7. In the power spacecraft subsystem, what components do the FOT monitor?

a) _____ confidence level

b) ____ batteries ____ solar arrays

8. In the attitude spacecraft subsystem, what components do the FOT monitor?

a) _____ confidence level

b) ____ gyros

9. In the command and data handling spacecraft subsystem, what components do the FOT monitor?

a) _____ confidence level

b) ____ science instruments ____ tape recorders

____ telemetry and command unit ____ memory

10. What are the two types of memory in a spacecraft??

a) _____ confidence level

b) ____ normal ____ block

12. Trace data flows within the spacecraft

a) _____ confidence level

b) ____ instrument->tcu->taperecorder

____ tcu->attenna

13. How does the tutor interface represent a hardware failure?

a) _____ confidence level

b) ____ objects ____ paths

14. How does the interface distinguish between hardware and software failures.

a) _____ confidence level

b) _____

15. Distinguish between the flow of data during TDRSS versus ground network support.

a) _____ confidence level

b) _____

16. Trace data flows within MSOCC and the TAC. Describe how a failure of a component affects data flows.

a) _____ confidence level

b) _____

17. Consider the support configuration panel, describe the components that will support the ground control and identify the activities that will be carried out during the scheduled support.

a) _____ confidence level

b) _____

GT-VITA Knowledge Questionnaire: GT-POCC Operations

Date: _____

Analyst: _____

1. What are the major pre-pass FOT tasks for a real-time support?

a) _____ confidence level

b) ____ AP init ____ TAC init ____ DCI

____ NCC request ODMs ____

2. What are the major on-pass FOT tasks for a real-time support?

a) _____ confidence level

b) ____ tape recorder dump ____ CSM load ____ CSM dump

____ monitor ground system ____ monitor spacecraft

3. What components of the ground system does the FOT monitor during a real-time support? What indicates a failure at a component?

a) _____ confidence level

b) ____ monitor MSOCC ____ monitor TAC

____ monitor AP ____ monitor NASCOM

c) _____ confidence level

d) ____ monitor MSOCC ____ monitor TAC

____ monitor AP ____ monitor NASCOM

4. What components of the spacecraft does the FOT monitor during a real-time support? What kinds of failures occur?

a) _____ confidence level

b) ___ AS ___ CDHS ___ PS ___ RFS

c) _____ confidence level

d) ___ AS ___ CDHS ___ PS ___ RFS

5. What steps are involved in a tape recorder dump?

a) _____ confidence level

b) ___ t/r "stdby" ___ t/r "playback" ___ t/r "record"

6. What steps are involved in a CSM load?

a) _____ confidence level

b) ___ CSM1 "load" ___ CSM1 "load file" ___ CSM1 "enable"

7. What steps are involved in a CSM dump?

a) _____ confidence level

b) ___ CSM1 "sump" ___ STOL VLDVLD

___ CSM1 "enable"

8. What are the steps involved in the forward and return link reacquisition after "loss of lock"?

a) _____ confidence level

b) ___ RFS ___ ACS ___ CDHS ___ PS

9 . What are the major post-pass FOT tasks for a real-time support?

a) _____ confidence level

b) ___ AP terminate ___ bookkeeping

APPENDIX D

GT-POCC OPERATIONS CHECKLIST

Major FOT Analyst Duties in GT-POCC:

Pre-Pass, On-Pass, and Post-Pass Checklists

An FOT analyst has three sets of duties: pre-pass, on-pass, and post-pass

Pre-Pass duties are to **configure** and **verify** ground support systems, and to perform appropriate **bookkeeping** before a real-time support begins.

On-Pass duties are to **monitor** data during the support, **command** the spacecraft if necessary, and to perform appropriate **bookkeeping** tasks.

Post-Pass duties are to **terminate** the Application Processor(AP) operation and to perform appropriate **bookkeeping** tasks.

Pre Pass: Configuration

The following activities should be carried out within ***two minutes*** before the acquisition of signal. The Applications Processor must be configured first.

1. Request to configure the Applications Processor (AP).

   Select the **MSOCC icon**
   Select the **Application Processor icon**
   Select the "**initialize**" menu item
   Click **Okay** button

2. Once the Applications Processoris initialized (you can see a message to this effect on the Events page), request to configure the Telemetry and Command Computer (TAC).

   Select the **MSOCC icon**
   Select the **Telemetry and Command Computer icon**
   Select the "**initialize**" menu item
   Click **Okay** button

Pre-Pass: Verification

The following activities should be carried out after configuration. These activities may be done in any order, provided they are complete before the acquisition of signal.

1. Verify communications with the Network Control Center (NCC).

   Select the **NCC icon**

Select the **"Request operation data messages [NCCHEK]"** menu item
Click **Okay** button

2. If this support is coherent, inhibit Doppler compensation.

   Select the **NCC icon**
   Select the **"Request Doppler compensation inhibition [DCI]"**
   Click **Okay** button

3. Verify GASP's command storage memory listing.

   Select **Other Actions** menu
   Select **"Pre-Pass Verification"** menu item
   Select **"Check CSM Listing [CSMVLD]"** menu item
   Click **Okay** button

4. Verify the AP software.

   Select **Other Actions** menu
   Select **"Pre-Pass Verification"** menu item
   Select **"Check Application Software [SOFTWARE]"** menu item
   Click **Okay** button

5. Verify GASP's support schedule.

   Select **Other Actions** menu
   Select **"Pre-Pass Verification"** menu item
   Select **"Check Support Schedule [SCHEDULE]"** menu item
   Click **Okay** button

Pre-Pass: Bookkeeping

Request pass plan page and fill in pre-pass information.

Select **Other Actions** menu
Select **"Bookkeeping"** menu item
In **Pass Plan Completion** menu,
    select **"Pass plan page [PASSPLAN]"** menu item
Click **Okay** button
In **Pass Plan Completion** menu,
    select **"Pre-pass information [PREPLAN]"** menu item
Click **Okay** button

On-Pass: Monitoring

The following monitoring tasks need to be managed throughout the support.

1. Monitor data flowing through MSOCC. Be sure that the components are not red nor the data flows are not red.

Select **MSOCC**

2. Monitor data arriving at the TAC. Watch the A and B channels to ensure that the TAC is operating properly. Be sure that neither the channels nor the data flows are red.

Select the **MSOCC icon**
Select the **TAC icon**

2. Monitor spacecraft health and safety data.

Select the **Spacecraft icon**
Select the **Power System icon**
check **voltage and cd ratios**
Select the **Radio Frequency icon**
check **transponders' agc and antenna angles**
Select **Command and Data Handling icon**
check **data flows** from instrument to tape recorder
select **memory** and check **normal and block memory**
Select **Attitude Control System icon**
check the **x, y, z gyro angles**

3. Monitor communications with other facilities. Watch the EVENTS page to ensure that you are receiving operation data messages (ODMs) from the Network Control Center every 20 seconds, and also watch for any other message from the DOCS, White Sands, TNC, or a ground network facility. Also, watch the EVENTS page to verify FOT analyst actions.

Select **Status Information** menu
Select **"System Events"** menu item

4. Run STOL procedures to check TDRSS and ERBE and SAGE science instruments parameters. Messages regarding the successful completion of these checks are shown on the EVENTS page.

Select **Other Actions** menu
Select **"STOL Procedures"** menu item
Select **"check subsystems during a TDRSS support [TDRSSCHK]"**
Click **Okay** button
Select **"check ERBE instrument [ERBECHK]"**
Click **Okay** button
Select **"check Sage instrument [SAGECHK]"**
Click **Okay** button

On-Pass: Bookkeeping

1. Request pass plan page and enter on-pass information.

    Select **Other Actions** menu
    Select **"Bookkeeping"** menu item
    In **Pass Plan Completion** menu,
        Select **"Pass plan page [PASSPLAN]"** menu item
    Click **Okay** button
    In **Pass Plan Completion** menu,
        Select **"On-pass information [ONPLAN]"** menu item
    Click **Okay** button

2. If a spacecraft parameter fails, create a spacecraft anomaly report.

    Select **Other Actions** menu
    Select **"Bookkeeping"** menu item
    In **Anomaly Report Completion** menu,
        Select **"Request spacecraft anomaly report [SPANOM]"**
    Click **Okay** button
    In **Anomaly Report Completion** menu,
        Select **"Fill in spacecraft anomaly [CSPANOM]"** menu item
    Select **Details** and select specific spacecraft parameter that failed
    Click **Okay** button

3. If a command storage memory fault occurs, create a command storage memory anomaly report.

    Select **Other Actions** menu
    Select **"Bookkeeping"** menu item
    In **Anomaly Report Completion** menu,
        Select **"Request CSM anomaly report [CSMANOM]"**
    Click **Okay** button
    In **Anomaly Report Completion** menu,
        Select **"Fill in CSM anomaly [CCSMANOM]"** menu item
    Select **Details** and select specific location of the fault
    Click **Okay** button

4. If signal lock is lost or a hardware or software failure occurs, create an events report.

    Select **Other Actions** menu
    Select **"Bookkeeping"** menu item
    In **Events Report Completion** menu,
        Select **"Request events report [EVREPORT]"**
    Click **Okay** button
    In **Event Report Completion** menu,
        Select **"Fill in event report [EVENT]"** menu item
    Select **Details** and select the specific component
    Click **Okay** button

Post-Pass: Termination

At the end of the support, you need to terminate the AP.

Select the **MSOCC icon**
Select the **AP icon**
Select the "**AP terminate**" menu item
Click **Okay** button

Post-Pass: Bookekeeping

Request pass plan page and fill in pre-pass information.

Select **Other Actions** menu
Select "**Bookkeeping**" menu item
In **Pass Plan Completion** menu,
    select "**Pass plan page [PASSPLAN]**" menu item
Click **Okay** button
In **Pass Plan Completion** menu,
    select "**Post-pass information [POSTPLAN]**" menu item
Click **Okay** button

With respect to your duties as an FOT analyst, there are three types of real-time supports: normal, TR dump, and CSM load. A normal support only requires that you perform routine configuration and verification, monitoring, and bookkeeping tasks. A TR (tape recorder) dump support requires that you command the spacecraft to play back (or "dump") data from one of GASP's onboard tape recorders in addition to your normal duties. A CSM (command storage memory) load support requires that you uplink and validate a command storage memory load in addition to your normal duties.

Commanding for a Tape Recorder Playback Event

You can find out from the pass plan which tape recorder is scheduled to playback its data.

1. Set the tape recorder to standby mode.

   Click either TR1STBY or TR2STBY on the command panel, depending on which tape recorder will be played back.

2. Set the tape recorder to playback mode.

   Click either TR1PBK or TR2PBK on the command panel, depending on which tape recorder will be played back.

3. When playback is complete (check the RUPS or TIPIT displays to see that both begin and end times are written in green; also the message "DONE: Playback complete" will appear on the events page), complete a data accountability report.

4. Set the tape recorder to record mode.

   Click either TR1REC or TR2REC on the command panel, depending on which tape recorder was played back.

Commanding a CSM Load Event

1. Disable CSM1.

   Click on the CSM1DIS button on the command panel.

2. Set CSM1 to load.

   Click on the CSM1LD button on the command panel.

3. Uplink the load file.

   Click the appropriate load button on the command panel.

4. When complete (check the Events page), dump the newly-loaded contents of CSM1's normal memory.

   Click the CSM1DMP button on the command panel.

6. When complete, (check the Events page), run the validation procedure to verify that the load was successful.

   Click the VLDVLD button on the command panel. A page will appear showing a list of commands from the spacecraft compared to the ground image. You can scroll down the window to see a command-by-command comparison of the load just sent. Check to make sure that "zero failures" occurred.

7. Enable CSM1 again.

   Click the CSM1EN button on the command panel.

REFERENCES

Adelman, L. (1991). Experiments, quasi-experiments, and case studies: A review of empirical methods for evaluating decision support systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 21(2), 293-301.

Anderson, J.R. (1988). The expert module. In M.C.Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R., Boyle, C.F., and Reiser, B.J. (1985). Intelligent tutoring systems. *Science,* 228(4698), 456-462.

Bailin, S.C., Moore, J.M., Hilberg, R.H. and Murphy, E.D. (1988). A logical model of cooperating rule-based systems in space mission ground support facilities. Paper prepared for NASA Goddard Space Flight Center, Computer Technology Associates, Inc.

Barr, A., Beard, M., and Atkinson, R.C. (1976). The computer as a tutorial laboratory: The Stanford BIP Project. *International Journal of Man-Machines Studies,* 8, 567-596.

Barr, A. and Feigenbaum, E.A. (1982). *The handbook of artificial intelligence,* Volume 2. Los Alto, CA: HeurisTech Press.

Bludworth, D. (1989). Verbal communications, NASA Goddard Space Flight Center.

Brown, J.S. (1985). Process versus product: a perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research,* 1, 179-201.

Bonar, J.G. and Cunningham, R. (1988). Bridge: Tutoring the programming process. In Psotka, J., Massey, L.D., and Tenney, Y.J. (Eds.), *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Bonar, J.G. and Weil, W. (1985). An informal programming language. Paper presented at the meeting Expert Systems in Government, October, Washington, D.C.

Brown, J.S., Burton, R.R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems.* New York: Academic Press.

Brown, J.S., Burton, R.R., and Zdydel, F. (1973). A model driven question-answering system for mixed-initiative computer-assisted instruction. *IEEE Transactions on Systems, Man, and Cybernetics,* 3(?), 248-257.

Burns, H.L. and Capps, C.G. (1988). Foundations of intelligent tutoring systems: An introduction. In M.C. Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Burns, H. and Parlett, J.W. (1991). The evolution of intelligent tutoring systems: Dimensions of design. In H. Burns, J.W. Parlett, and C.L. Redfield (Eds.), *Intelligent tutoring systems: Evolution in design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Burns, H., Parlett, J.W., and Redfield, C.L. (Eds.) (1991). *Intelligent tutoring systems: Evolution in design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Burton, R.R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Burton, R.R. (1988). The environment module of intelligent tutoring systems. In M.C. Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Burton, R.R. and Brown, J.S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Bushman, J.B., Mitchell, C.M., Jones, P.M. and Rubin, K.S. (1989). ALLY: An operator's associate model for cooperative supervisory control situations. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, November 14-19, Cambridge, MA.

Carbonell, J.R. (1970). AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4), 190-202.

Carr, B. and Goldstein, L.P. (1977). Overlays: A theory of modeling for computer-aided instruction. *AI Lab Memo 406 (Logo Memo 40)*. Massachusetts Institute of Technology, Cambridge, MA.

Chan, T. and Baskin, A.B. (1990). Learning companion systems. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: At the crossroad of artificial intelligence and education*. Norwood, NJ: Ablex Publishing.

Chronister, J.A. (1990). A domain-indepedent framework for structuring knowledge in the OFMspert Architecture. M.S. thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Geogia Institute of Technology, Altanta, GA.

Chu, R. W. and Jones, P. M. (1990a). Prototypical satellite ground control tasks and problems. CHMSR, School of ISyE, Georgia Institute of Technology, Atlanta, GA, unpublished.

Chu, R. W. and Jones, P. M. (1990b). GT-POCC scenarios and session definitions. CHMSR, School of ISyE, Georgia Institute of Technology, Atlanta, GA, unpublished.

Chu, R.W., Jones, P.M., and Mitchell, C.M. (1990). The Georgia Tech Payload Operations Control Center simulation: Design and implementation in C++. *Proceedings of the SCS Multiconference on Object-Oriented Simulation* , January 23-25, Anaheim, CA.

Clancey, W.J. (1986). From GUIDON to NEOMYCIN and HERACLES in twenty short lessons: ONR final report 1979-1985. *AI Magazine*, 7(3), 40-60.

Clancey, W.J. (1987). *Knowledge-based tutoring: The GUIDON program*. Cambridge, MA: The MIT Press.

Clymer, A.B. (1980). Simulation for training and decision-making in large-scale control systems. Part I: Types of training simulators. *Simulation*, 35(8), 39-41.

Fath, J. L., Mitchell, C.M., and Govindaraj, T. (1990). An ICAI architecture for troubleshooting in complex, dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 20(3), 537-558.

Fink, P.K. (1991). The role of domain knowledge in the design of an intelligent tutoring system. In H. Burns, J.W. Parlett, and C.L. Redfield (Eds.), *Intelligent tutoring systems: Evolution in design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Fox, B.A. (1988a). Interaction as a diagnostic resource in tutoring. *Report No. 88-3*, Department of Linguistics and Institute of Cognitive Science, University of Colorado, Boulder, CO.

Fox, B.A. (1988b). Cognitive and interactional aspects of correction in tutoring. *Report No. 88-2*, Department of Linguistics and Institute of Cognitive Science, University of Colorado, Boulder, CO.

Frasson, C. and Gauthier, G. (Eds.) (1990). *Intelligent Tutoring Systems: At the crossroad of artificial intelligence and education*. Norwood, NJ: Ablex Publishing .

Genesereth, M.R. (1982). The role of plans in intelligent teaching systems. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Gentner, D.R. (1979). Toward an intelligent computer tutor. In H. O'Neil (Ed.), *Procedures for instructional systems development*. New York: Academic Press.

Gibbons, J.D. (1985). *Nonparametric statistical inference* (second edition). New York: Marcel Dekker.

Goldstein, I.L. (1986). *Training in Organizations: Needs Assessment, Development, and Evaluation*. Pacific Grove, CA: Brooks/Cole Publishing.

Goldstein, I.P. (1980). Developing a computational representation for problem solving skills. In D. Tuma and F. Reif (Eds.), *Problem solving and education: Issues in teaching and research*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Goldstein, I.P. (1982). The genetic graph: A representation for the evolution of procedural knowledge. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Goodstein, L.P., Andersen, H.B., and Olsen, S.E. (Eds.) (1988). *Tasks, errors and mental models*. London: Taylor & Francis.

Govindaraj, T. (1987). Qualitative approximation methodology for modeling and simulation of large dynamic systems: Applications to a marine powerplant. *IEEE Transactions on Systems, Man, and Cybernetics,* 17(6), 937-955.

Hancock, P.A. and Pierce, J.O.(1984). Toward an attentional theory of performance under stress: Evidence from studies fo vigilance in heat and cold. In A. Mital (ed.), *Trends in ergonomics/human factors I.* New York: North Holland.

Halff, H.M. (1988). Curriculum and instruction in automated tutors. In M.C. Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Harris (in progress). M.S. thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Geogia Institute of Technology, Altanta, GA.

Hollan, H.D., Hutchins, E.L., and Weitzman, L. (1984). STEAMER: an interactive inspectable simulation-based training system. *AI Magazine,* 5(2), 15-27.

Johnson, W.B. (1988). Developing expert system knowledge bases in technical training environments. In J. Psotka, L.D. Massey, and Y.J. Tenney (Eds.), *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Johnson, W.B. (1988). Pragmatic considerations in research, development, and implementation of intelligent tutoring systems. In M.C. Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Johnson, W.L., and Soloway, E.M. (1985). PROUST: an automatic debugger for Pascal programs. *Byte,* 10(4), 179-190.

Jones, P.M. (1991). Human-computer cooperative problem solving in supervisory control. Doctoral thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Geogia Institute of Technology, Altanta, GA.

Jones, P. M., Chu, R. W. and Mitchell, C. M. (1990a). Knowledge requirements for the flight operations team analyst in the Goddard Space Flight Center Advanced POCC. CHMSR, School of ISyE,Georgia Institute of Technology, Atlanta, GA, unpublished.

Jones, P.M., Mitchell, C.M. and Rubin, K.S.(1990b). Validation of intent inferencing by a model-based operator's associate. *International Journal of Man-Machine Systems,* 33, 177-202.

Kearsley, G. (1984). *Training and technology: A handbook for HRD professionals.* Reading, MA: Addison-Wesley.

Kieras, D.E. (1988). What mental models should be taught: Choosing instructional content for complex engineered systems. In Psokta, J., Massey, L.D. and Mutter, S.A. (Eds.), *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Kimball, R. (1982). A self-improving tutor for symbolic integration. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems.* New York: Academic Press.

Kirpatrick, D.L. (1959, 1960). Techniques for evaluating training programs. *Journal of the American Society of Training Directors*, 13, 3-9, 21-26; 14, 13-18, 28-32.

Kyllonen, P.C. and Shute, V.J. (1989). A taxonomy of learning skills. In P. Ackerman, R. Sternberg, and R. Glaser (Eds.), *Learning and individual differences*. San Francisco, CA: W.H.Freeman.

Leplat, J. (1988). Task complexity in work situations. In L.P. Goodstein, H.B. Andersen, and S.E. Olsen (Eds.), *Tasks, errors and mental models*. London: Taylor & Francis.

Lesgold, A. (1988). Toward a theory of curriculum for use in designing intelligent instructional systems. In H. Mandl and A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Lesgold, A., Lajoie, S., Bunzo, M. and Eggan, G. (1988). SHERLOCK: A coached practice environment for an electronic troubleshooting job. In Larkin, J., Chabay, R. and Scheftic, C. (Eds.), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Mandl, H. and Lesgold, A. (Eds.) (1987). *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Massey, L.D., Bruin, J., and Roberts, B. (1988). A training system for system maintenance. In J. Psotka, L.D. Massey, and Y.J. Tenney (Eds.), *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.

McDonald, J. (1981). The EXCHECK CAI system. In Suppes, P. (Ed.), *University-level computer-assisted instruction at Stanford: 1968-1980*. Institute for Mathematical Studies in the Social Sciences, Stanford University, Stanford, CA.

Miller, M.L. (1982). A structured planning and debugging environment for elementary programming. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Miller, R.A. (1985). A system approach to modelling discrete control performance. In Rouse, W.B. (Ed.), *Advances in Man-Machines System Research*, Volume 2. Greenwich, CN: JAI Press.

Mitchell, C.M. (1991). Personal communications, Georgia Institute of Technology, Atlanta, GA.

Mitchell, C.M. (1987). GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(4), 553-572.

Mitchell, C.M. and Govindaraj, T. (1989). A tutor for satellite system operators. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, November 14-19, Cambridge, MA.

O'Shea, T. (1982). A self-improving quadratic tutor. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. New York: Academic Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Polson, M.C. and Richardson, J.J. (Eds.), *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rasmussen, J. (1986). *Information processing and human-machine interactions: An approach to cognitive engineering*. New York: North-Holland.

Rasmussen, J. and Lind, M. (1981). Coping with complexity. *Ris0 Report Ris0-M-2293*, Ris0 National Laboratory, Roskilde, Denmark.

Regian, J.W. (1991). Representing and teaching high performance tasks within intelligent tuttoring systems. In H. Burns, J.W. Parlett, and C.L. Redfield (Eds.), *Intelligent tutoring systems: Evolution in design*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Regian, J.W. and Schneider, W (1986). Assessment procedures for predicting and optimizing skill acquisition. Paper presented at ETS conference on diagnostic monitoring, July, Princton, NJ.

Reiser, B.J., Anderson, J.R., and Farrell, R.B. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. *Proceedings of the International Conference on Artificial Intelligence-85*, 1, 8-14. Los Alto, CA: Morgan Kaufmann.

Roth E.M., Bennett, K.B. and Woods, D.D. (1987). Human interaction with an "intelligent" machine. *International Journal of Man-Machine Systems*, 27(5-6), 479-525.

Rubin, K.S., Jones, P.M. and Mitchell, C.M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Man-Machine Systems*, 18(4), 618-637.

Sanberg, J.A.C. (1987). *The Third International Conference on Artificial Intelligence and Education*. AICOM, 0, 51-53.

Self, J.A. (1990). Bypassing the Intractable problem of student modeling. In C. Frasson, and G. Gauthier (Eds.), *Intelligent Tutoring Systems: At the crossroad of artificial intelligence and education*. Norwood, NJ: Ablex Publishing .

Schneider, W. (1985). Towards a model of attention and the development of automaticity. In M.I. Posner and O.S. Marin (Eds.), *Attention and performance XI* . Lawrence Erlbaum Associates, Hillsdale, NJ.

Sheridan, T.B. (1976). Toward a general model of supervisory control. In T.B. Sheridan, and G. Johannsen (Eds.), *Monitoring behavior and supervisory control*. New York: Plenum Press.

Sleeman, D.H. (1987). PIXIE: A shell for developing intelligent tutoring systems. In R. Lawler and M. Yazdani (Eds), *AI and education: Learning environments and intelligent tutoring systems*. Norwood, NJ: Ablex Publishing.

Sleeman, D.H. and Brown, J.S. (Eds.) (1982). *Intelligent tutoring systems*. New York: Academic Press.

Spangenberg, R.W. (1976). Selection of simulation as an instructional medium. *Proceedings of First International Learning Technology Conference and Exposition on Applied Learning Technology, Vol IV: Future of Simulators in Skills Training,* Society for Applied Learning Technology.

Stevens, A., Collins, A., and Goldin, S.E. (1982). Misconceptions in students' understanding. In D. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems.* New York: Academic Press.

Suchman, L.A. (1987). *Plans and situations actions: The problem of human-machine communication.* Cambridge: Cambridge University Press.

Swigger, K.M. (1991). Managing communication knowledge. In H. Burns, J.W. Parlett, and C.L. Redfield (Eds.), *Intelligent tutoring systems: Evolution in design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

TAE Plus (1990). Overview, Version 4.1, NASA Goddard Space Flight Center.

Thompson, P.W. (1987). Mathematical microworlds and intelligent computer-assisted instruction. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods.* Reading, MA: Addison-Wesley.

Towne, D.M. and Munro, A. (1988). The intelligent maintenance training system. In Psotka, J., Massey, L.D., and Tenney, Y.J. (Eds.), *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum Associates.

VanLehn, K. (1988). Student Modelling. In M.C. Polson and J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Verfurth, S.C. (1991). Modeling the pilots and constructing an intent inferencer for a Boeing 727 cockpit. M.S. thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Geogia Institute of Technology, Altanta, GA.

Vasandani, V., Govindaraj, T., and Mitchell, C.M. (1989). An intelligent tutor for diagnostic problem solving in complex dynamic systems. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, November 14-19, Cambridge, MA.

Vasandani, V. and Govindaraj, T (1990). Knowledge representation and human-computer interaction in an intelligent tutor for diagnostic problem solving. *Proceedings of the 1990 IEEE International Conference on Systems, Man, and Cybernetics*, November 4-7, Los Angeles, CA.

Vasandani, V. (1991). Intelligent tutoring for diagnostic problem solving in complex dynamic systems. Doctoral thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Geogia Institute of Technology, Altanta, GA.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann.

Wickens, C.D. (1984). *Engineering psychology and human performance*. Columbus, OH: Merrill.

White, B.Y. and Frederiksen, J.R. (1987). Qualitative models and intelligent learning environments. In R. Lawler and M. Yazdani (Eds.), *AI and education: Learning environments and intelligent tutoring systems*.. Norwood, NJ: Ablex Publishing.

Williams, M.D., Hollan, J.D., and Stevens, A.L. (1981). An Overview of STEAMER: An advanced computer-assisted instruction system for propulsion engineering. *Behavior Research Methods and Instrumentation*, 13(2), 85-90.

Woods, D.D. (1986a). Cognitive technologies: The design of joint human-machine cognitive systems. *The AI Magazine*, 6(4), 86-92.

Woods, D.D. (1986b). Paradigms for intelligent decision support. In E. Hollnagel, G. Mancini, and D.D. Woods (Eds.), *Intelligent decision support in process environments*. New York: Springer-Verlag.

Woods, D.D. (1987). Commentary: Cognitive engineering in complex and dynamic worlds. *International Journal of Man-Machine Systems*, 27(5-6), 571-585.

Woods, D.D. (1988). Coping with complexity: The psychology of human behaviour in complex systems. In L.P. Goodstein, H.B. Andersen, and S.E. Olsen (Eds.), *Tasks, errors and mental models*. London: Taylor & Francis.

Woods, D.D. and Roth, E.M. (1987). Cognitive systems engineering. In M. Hollander (Ed.), *Handbook of Human-Computer Interaction*. New York: Springer-Verlag.

Woolf, B.P. (1986). Teaching a complex industrial process. *COINS Technical Report 86-24*, Computer and Information Science, University of Massachusetts, Amherst, MA.

Woolf, B.P. and McDonald, D.D. (1984) Building a computer tutor: Design issues. *IEEE Computer*, 17(9), 61-73.

VITA

Rose Wan-Mui Chu was born in 1961 in Hong Kong. After graduating from high school in Malaysia, she began college in the University of Montevallo, Alabama, in January, 1980. She eventually received the Bachelor of Industrial Engineering degree from the Auburn University, Alabama, in August, 1983. She graduated with highest honor and was also the recipient of the Outstanding Graduate for the School of Engineering. Upon graduation, she began graduate studies in industrial engineering at Auburn University and received the Master of Science degree in January, 1987.